Martin Thorsen Ranang

Open-Domain Word-Level Interpretation of Norwegian

Towards a General Encyclopedic
Question-Answering System for Norwegian

Thesis for the degree of Philosophiae Doctor

Trondheim, February 2010

Norwegian University of Science and Technology
Faculty of Information Technology, Mathematics
and Electrical Engineering
Department of Computer and Information Science

NTNU
Norwegian University of Science and Technology

Thesis for the degree of Philosophiae Doctor

Faculty of Information Technology, Mathematics and Electrical Engineering
Department of Computer and Information Science

ISBN 978-82-471-1973-0 (printed ver.)
ISBN 978-82-471-1974-7 (electronic ver.)
ISSN 1503-8181

Doctoral theses at NTNU, 2010:11

Printed by NTNU-trykk

For Johanna

Contents

List of Figures

List of Tables

List of Algorithms

Preface

This work is submitted to the Norwegian University of Science and Technology in partial fulfillment of the requirements for the degree *Philosophiae Doctor*. This work has been completed at the Department of Computer and Information Science at Norwegian University of Science and Technology, Trondheim. My main advisor has been Associate Professor *Tore Amble*, while my co-advisors have been Professor *Torbjørn Nordgård* and Professor *Björn Gambäck*.

Martin Thorsen Ranang

Publications

Some ideas and figures presented herein have appeared previously in the following publications:

Nordgård, Torbjørn, Martin Thorsen Ranang, and Jostein Ven. 2005. An approach to automatic text production in electronic medical record systems. In *Proceedings of the 9th International Conference on Knowledge-Based Intelligent Information and Engineering Systems (KES 2005)*, ed. Rajiv Khosla, Robert J. Howlett, and Lakhmi C. Jain, vol. 3683 of *Lecture Notes in Artificial Intelligence*, 1187–1194. Melbourne, Australia: Springer-Verlag, Berlin, Heidelberg.

Sætre, Rune, Martin Thorsen Ranang, Tonje S. Steigedal, Kamilla Stunes, Kristine Misund, Liv Thommesen, and Astrid Lægreid. 2007. WebProt: Online mining and annotation of biomedical literature using Google. In *Advanced computational methods for biocomputing and bioimaging*, ed. Tuan D. Pham, Hong Yan, and Denis I. Crane. Hauppauge, New York, USA: Nova Science Publishers.

Sætre, Rune, Amund Tveit, Martin Thorsen Ranang, Tonje S. Steigedal, Liv Thommesen, Kamilla Stunes, and Astrid Lægreid. 2005. gProt: Annotating protein interactions using Google and Gene Ontology. In *Proceedings of the 9th International Conference on Knowledge-Based Intelligent Information and Engineering Systems (KES 2005)*, ed. Rajiv Khosla, Robert J. Howlett, and Lakhmi C. Jain, vol. 3683 of *Lecture Notes in Artificial Intelligence*, 1195–1203. Melbourne, Australia: Springer-Verlag, Berlin, Heidelberg.

Acknowledgments

There are many people I want to thank, that in some way or another have helped me finish this dissertation.

First of all I would like to thank my advisers, *Tore Amble*, *Torbjørn Nordgård*, and *Björn Gambäck* for all their patience and advices. I am also deeply grateful to *Mila Dimitrova-Vulchanova* and *Sindre Bjørnar Norås* for enabling me to perform the tests for evaluating the quality of the mapping method.

I would also like to thank *Kristin Melum Eide* for explaining and discussing parts of the linguistic theory with me. I am also grateful for the work she and *Tore Amble* did together, carefully crafting the new grammar for TUC, based on *Norsk referansegrammatikk* (Faarlund et al. 1997).

Furthermore, I would like to thank *Genevieve Gorrell* for asking a couple of "right questions" at the right time; *Tore Bruland* for fruitful discussions and valuable tips; *Janne Bondi Johannessen* for encouraging conversations; *Helge Dyvik* for an interesting and inspiring discussion; *Magnus Lie Hetland* for answering my questions about just about anything; *Marte Fodstad* for being a good friend and giving me great advise in a pressured situation; and *Olav Mørkrid* for teaching me the joys of assembly programming when I was about 11 years old, and thereby gave me an introduction to the wonderful world of computers and programming.

The following additional friends and colleagues made my stay in Trondheim a period of my life I will remember as exciting and enjoyable: *Rolv Inge Seehuus*, *Per Kristian Lehre*, *Rune Sætre*, and *May Elisabeth Bayegan*. Thank you all.

I am forever grateful to *Harald Nordgård-Hansen* and *Martin Kermit* for introducing me to the joys of problem solving and inspiring me to pursue academical work.

Without *Jörg Cassens*' patience, many helpful advices, and interesting late-night discussions, I would probably never have made it. So, thank you.

I would also like to thank *Einar Dehli* and *Kjetil Moløkken-Østvold* at my current employer, Conceptos IT Development AS, and *Trond Heier* at my previous employer, Redpill Linpro AS, for generously providing time for me to finish this work.

I would like to thank both Silje, my wonderful wife, and Johanna, my fantastic and inspiring daughter, for their patience and for letting me finish this work.

Finally, I would like to thank *Jo Henning Myhrvang*, my parents, my great cousin *Axel Ranang Gustavsen*, and the rest of my family for their repeated encouragements, great support, and patience.

So, once again, to all of you, and anyone I might have forgotten: thank you! Your support was important to me.

Abstract

No large-scale, open-domain semantic resource for Norwegian, with a rich number of semantic relations currently exists. The existing semantic resources for Norwegian are either limited in size and/or incompatible with the *de facto* standard resources used for Natural Language Processing for English. Both current and future cultural, technological, economical, and educational consequences caused by the scarcity of advanced Norwegian language-technological solutions and resources has been widely acknowledged (Simonsen 2005; Norwegian Language Council 2005; Norwegian Ministry of Culture and Church Affairs 2008).

This dissertation presents (1) a novel method that consists of a model and several algorithms for automatically mapping content words from a non-English source language to (a power set of) WordNet (Miller 1995; Fellbaum 1998c) senses with average precision of up to 92.1 % and coverage of up to 36.5 %. Because an important feature of the method is its ability to correctly handle compounds, this dissertation also presents (2) a practical implementation, including algorithms and a grammar, of a program for automatically analyzing Norwegian compounds. This work also shows (3) how Verto, an implementation of the model and algorithms, is used to create Ordnett, a large-scale, open-domain lexical-semantic resource for Norwegian with a rich number of semantic relations. Finally, this work argues that the new method and automatically generated resource makes it possible to build large-scale open-domain Natural Language Understanding systems, that offer both wide coverage and deep analyses, for Norwegian texts. This is done by showing (4) how Ordnett can be used in an open-domain question answering system that automatically extracts and acquires knowledge from Norwegian encyclopedic articles and uses the acquired knowledge to answer questions formulated in natural language by its users. The open-domain question answering system, named TUClopedia, is based on *The Understanding Computer* (Amble 2003) which has previously been successfully applied to narrow domains.

—Most, if not all, high-end natural language processing applications—from the earliest, machine translation, to the latest, question answering and text summarization—stand to benefit from being able to use text meaning.

Nirenburg and Raskin (2004)

Introduction

1

The languages shared and spoken by a nation's citizens are woven into their culture, psychology, education and politics (Lambert 1972; Joseph 2004). An important aspect of protecting and evolving each nation's distinctive features in the age of globalization is to stimulate use of their languages.

At the same time, much of today's most advanced language technology is only available for the English language. For example, the question-answering (QA) system Powerset is able to search, read, and answer questions about Wikipedia articles, but only in English (Converse et al. 2008).[1] Besides, when advanced language technology is also made available for other, non-English languages, the natural tendency is to prioritize languages with many speakers, or of great military and/or economical significance.

Even though the importance of language learning and language technology for non-English languages in Europe has been recognized (Laver and Roukens 1996), a tendency of English becoming a *lingua franca* for international communication—in particular within institutions of the European Union (EU)—has been observed (van Els 2001).

Norwegian is an example of a language with relatively few speakers; Norway has a population of 4,799,252, of which 303,000 are foreign citizens (Statistics Norway 2009). Discussing how to guard the Norwegian language's survival and evolution in an age where small languages are increasingly pressured in the context of global language development, both Simonsen (2005; p. 269) and the Norwegian Language Council (2005; pp. 118–137) advised—among other measures—that Norwegian versions of advanced language technology products must be made available to stimulate the use of Norwegian.

The Norwegian Council of State later approved a general language-political proposition by the Norwegian Ministry of Culture and Church

1 Powerset, `http://www.powerset.com/`. Accessed May 22, 2009.

Affairs (2008; pp. 134–137) that emphasizes that the Norwegian language and culture will be strengthened by making available advanced language-technological solutions. The proposition also stressed the importance of establishing a Norwegian Human Language Technology (HLT) resource collection.[2] Creating and making such resources available has major consequences within several areas, including research, education, business, and culture. Furthermore, it makes it easier—or even viable—to develop advanced Norwegian language technology solutions such as Natural Language Understanding (NLU) systems or systems like the QA system mentioned above.

Practical Problem

As we can see, the relative scarcity of advanced Norwegian language-technological solutions and resources—compared to the situation for other, larger languages—has been widely acknowledged.

1.1 Practical Problem

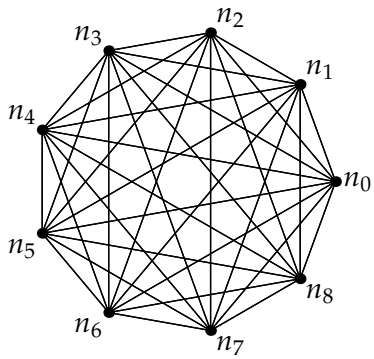
No large-scale general lexical-semantic resources, that both cover a broad domain of discourse and contain a rich number of semantic relations, exist for the Norwegian language. One of the consequences is that no large-scale open-domain NLU system that offers both a wide coverage and deep analyses for Norwegian texts can be built.

There are mainly two different approaches to developing such a semantic resource. One is to develop a stand-alone resource, and the other one is to create a linkage to an existing resource that can be used as a common resource/representation, or hub. Figure 1.1a on the facing page shows a situation where one tries to enable interoperability between each resource n_i in a set of resources $\{n_0, \ldots, n_8\}$ with every other resource in the set. The amount of work required to manage this is $O(n^2)$. If, on the other hand, a common ontology is used, symbolized as c in Figure 1.1b, the amount of work is reduced to $O(n)$.

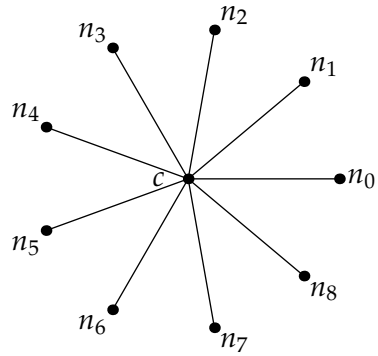
The semantic resource created by the method presented by Dyvik (2004) distinguishes between senses of words, and contains semantic relations like synonymy, hypernymy/hyponymy, and the less standard related word (for senses for which neither of the former relations hold).

First of all, Dyvik's approach creates a resource without any links, or interfaces, to other semantic resources. This means, that even though the semantic-mirrors method is automatic—and can easily generate similar new semantic resources based on new corpora as input—the cost of integrating it with n other semantic resources and applications is likely to be $O(n^2)$.

2 The Norwegian Human Language Technology Resource Collection, `http://www.spraakbanken.uib.no/`. Accessed May 22, 2009.



(a) Fully connected mesh topology. Without any common hub, the effort needed to make n resources interoperable will be $T(n) = n \cdot (n-1) = O(n^2)$, because for each one of the n resources mappings must be created to each of the $n-1$ other resources.



(b) Star topology. Given a common hub, c, the effort needed to align n resources will be $T(n) = 2n = O(n)$, because mappings must be created from each of the n resources to the common resource and—depending on the needs of the application—possibly a mapping from the hub to each resource (hence $2n$).

FIGURE 1.1: Approaches to integrating semantic knowledge sources.

Secondly, the number of semantic relations made available with Dyvik's approach is rather modest. In comparison, the *de facto* standard lexical-semantic resource WordNet (Miller 1995; Fellbaum 1998c) contains the semantic relations synonymy and antonymy for nouns, verbs, adjectives, and adverbs; hypernymy/hyponymy for nouns and verbs; holonymy/meronymy for nouns; entailment, cause to, and verb group for verbs; and similarity for adjectives. Table 4.4 on page 80 shows an overview of the semantic relationships defined in WordNet.

Research Problem

The resource automatically generated from the method described by Nygaard (2006) contains synonymy and hypernymy/hyponymy relations between *words*. However, it does not distinguish between *senses* of words. As also pointed out by Nygaard (2006; pp. 62–66), this may lead to erroneous interpretations and conclusions. For example, even if an "organ" is-a-kind-of "musical instrument", and a "heart" is-a-kind-of "organ", a "heart" is *not* a kind of "musical instrument". The reason is that the hypernymy relations are defined for different *senses* of those words. Hence, it does not seem meaningful to use the produced resource even for simple reasoning.

The NLU system *The Understanding Computer* (TUC) by Amble (2003) has been successfully applied to several narrow domains that will be presented in Section 2.7.1 (see, for example, Amble 2000; Bruland 2002; Sætre 2006). However, these versions of TUC's semantic network reflect the narrowness of their domains.

Also, just as for Dyvik's approach, the resource generated by Nygaard's approach and the semantic networks used in different versions of TUC are decoupled from other semantic resources and applications.

1.2 Research Problem

There are four central problems with the existing approaches to developing lexical semantic resources for Norwegian that the systems mentioned above suffer from:

1 Per definition, narrow-domain ontologies cannot be used for broad-domain Natural Language Processing (NLP).

2 Separately developed ontologies are extremely costly to develop, both with respect to the initial development and to the long-term maintenance. Systems based on shared, or aligned, ontologies can benefit from collective experiences and improvements.

3 While several new semantic resources are being designed to be compatible with existing ones, separately developed noncompatible resources cannot easily benefit from such synergism.

4 It seems that the semantic resources for Norwegian, produced by the current methods, lack a number of interesting semantic relations.

On the other hand, one should note that there are at least two reasons for developing ontologies in-house. One reason is that one can tailor one's ontology to better suit both the application domain and the methodologies used in the application. Another reason is that a suitable, freely available resource might not exist when the need for it arises.

Thus, the research problem is twofold. Firstly, no broad-domain semantic resource for Norwegian with a rich number of semantic relations currently exists, and secondly, the existing semantic resources for Norwegian are either limited in size and/or incompatible with the *de facto* standard resources used for NLP for English.

1.3 Research Questions

The research problem discussed above lead to the following research questions:

Q1 Is it possible to develop a method for automatically building a broad-domain, general semantic resource for Norwegian that is compatible with existing freely available, widely used, English semantic resources?

Q2 What restrictions apply to the method? What resources are prerequisite?

Q3 For each of the lexical categories nouns, verbs, adjectives, and adverbs, how large fraction of the words in each class does the method work for?

Q4 How well does the method avoid mapping words in the source language to senses in the target language that carry meanings that are not covered by the source words?

Q5 Can the method handle (single word) compounds that occur frequently in Norwegian, but more seldom in English?

Q6 How may the resulting semantic resource be useful for different areas of NLP research and development?

Q7 Can the method be applied to other languages, and if so, to which ones?

Section 6.3 summarizes the corresponding answers.

1.4 Research Solution

Research Solution

In this dissertation I present (1) a method I have developed that consists of a model and several algorithms for automatically mapping content words from a non-English source language to (a power set of) WordNet (Miller 1995; Fellbaum 1998c) senses.

An important feature of the method is its ability to handle (single-word) compounds. Therefore, I also present (2) a practical implementation, including algorithms and a grammar, of a program for automatically analyzing Norwegian compounds.

I will also show (3) how Verto, an implementation of the model and algorithms, is used to create Ordnett, a large-scale, open-domain lexical-semantic resource for Norwegian with a rich number of semantic relations.[3]

Finally, I argue that my method and automatically generated resource makes it possible to build large-scale open-domain NLU systems, that offer both wide coverage and deep analyses, for Norwegian texts. I do this by showing (4) how Ordnett can be used in an open-domain question answering (open-domain QA) system that automatically extracts and acquires knowledge from Norwegian encyclopedic articles and uses the acquired knowledge to answer questions formulated in natural language by its users. The system, named TUClopedia, is based on TUC (Amble 2003) which has previously has been successfully applied to narrow domains.

1.5 Roadmap

Chapter 2 introduces central concepts, background theory, and research related to the rest of the dissertation. It starts with an introduction of topics like ontologies, semantic networks, and lexical semantics. This is followed by an overview of some well-known lexical semantic resources for non-English languages.

Because the ability to handle compounds is an important feature of the method for automatically mapping content words from a non-English language to WordNet senses, Chapter 3 presents a practical implementation, including a complementary specification, of an algorithm for automatic analysis and treatment of Norwegian compounds. At the end of the chapter, the performance of the compound-word analyzer is evaluated.

3 The mapping framework is named Verto, from the Latin verb *verto* with several meanings, including "to turn around", "to translate", and "to turn out" (Woodhouse 1982), all describing the nature of the framework.

The novel method for automatically mapping both simplex and compound content words in a non-English source language to WordNet senses is presented in Chapter 4. Section 4.2 attempts to convey the intuition behind the method, while sections 4.3–4.8 provide a more formal presentation of the model and algorithms behind the method. Section 4.9 shows how the method is implemented as a framework named Verto, while the final sections evaluate the implemented method.

The experiments conducted to evaluate Verto are also used to create a novel lexical-semantic resource named Ordnett. The new lexical-semantic resource contains mappings from Norwegian words to WordNet senses. As an example of a Norwegian, advanced language-technological application made viable by the resource generated by my method Chapter 5 presents, TUClopedia, a proof-of-concept prototype of a Norwegian open-domain QA system. TUClopedia shows how Ordnett can be used in a system that automatically extracts and acquires knowledge from a Norwegian encyclopedia and answer users' questions, formulated in natural language, based on the acquired knowledge. Thus, TUClopedia constitutes an NLP system that offers both wide coverage and deep analyses. Section 5.4 presents an overview of the TUClopedia system, while sections 5.7 and 5.8 present and discuss some of the experiences made with the system so far. The final sections also discuss some of the difficulties that must be overcome before TUClopedia is finished.

Chapter 6 provides further analyses and discussion of the results from chapters 3–5, especially for topics that span more than what is covered in each of the previous chapters in isolation. Particularly, Section 6.3 provides answers to the research questions and discusses the contributions made by this dissertation. Finally, a discussion of possible future, related research is given.

1.6 Who Did What?

All the work and research presented in chapters 3 and 4 was done by the author.

In Chapter 5 contributions were made both by *Tore Amble* and the author. Amble has been developing TUC since the early 1990's. He also created LexTUC, a generalization of TUC that constitutes important parts of the TUClopedia system. LexTUC manifests itself as the Parser, Semantic interpreter, Reasoning engine, and Semantic network components of TUClopedia, as shown in the overview in Figure 5.1. The author implemented the other parts of TUClopedia, including the integration of LexTUC in the overall system.

The implementation of the algorithms and framework presented in chapters 3 and 4, and all of TUClopedia—except the parts contributed by Amble as described above—amounts to 20,982 lines of Python[4] code and 1,695 lines of ANSI C code written by the author. The LexTUC implementation, by Amble, amounts to 9,253 lines of Prolog code.[5]

Who Did What?

1.6.1 *The Publications*

This section describes my contributions to the publications listed on page xxi.

The article by Nordgård, Ranang, and Ven (2005) was on the whole written by *Torbjørn Nordgård*. I contributed comments and discussions, especially about the scaling aspects of the presented approach. I also formalized the presentation of the cost algorithm (Nordgård et al. 2005; p. 1191).

For the article by Sætre, Tveit, Ranang, Steigedal, Thommesen, Stunes, and Lægreid (2005), my contribution was the algorithm for multiple-pattern approximate sequence matching, and its implementation, the essential part of Step 4, "Parsing", of the gProt approach (Sætre et al. 2005; p.1197–1198).

The same algorithm was used by Sætre, Ranang, Steigedal, Stunes, Misund, Thommesen, and Lægreid (2007), and I explained the workings of the algorithm in that book chapter (Sætre et al. 2007; pp.197–199).

A slightly improved version of the multiple-pattern approximate sequence matching algorithm is presented in more detail in Section 5.5 of this dissertation.

4 The Python programming language, `http://www.python.org/`. Accessed May 22, 2009.
5 The author made a few minute modifications to LexTUC, but they are insignificant compared to Amble's contribution.

—If I have seen a little further it is by standing on the shoulders of Giants.

Isaac Newton, in a letter to Robert Hooke dated 5th of February 1676

Background and Related Research

2

To process and interpret textual input written in a particular language a Natural Language Understanding (NLU) system must have some knowledge about both the language and the world that input statements refer to. A natural division of such knowledge into parts, yields

- *lexical knowledge* about the different words' lexical categories[1] and morphological features;
- *grammatical knowledge* about which combinations of words and phrases constitute grammatical sentences in the language under consideration; and
- *semantic knowledge* about how the meaning of words and concepts relate to the meaning of other words and concepts.

Such resources, containing the above-mentioned kinds of knowledge are often called lexicons, grammars, and ontologies (or semantic networks), respectively. It should be noted that sometimes the different kinds of knowledge overlap.

In their comprehensive work on ontological semantics, Nirenburg and Raskin (2004; p. 88) note that

> It is practically and technologically impossible to operate with elements of the outside world as the realm of meaning for natural language elements. Therefore, if one wants to retain the capability of representing and manipulating meaning, a tangible set of meaning elements must be found to substitute for the entities in the outside world.

There are several ways of incorporating such knowledge in a computer program. For example, one can design algorithms so that they

1 Lexical categories are often also referred to as part of speech (POS) and word classes.

implicitly manifest an encoding of such knowledge, or one can separate the knowledge from the algorithms by making it accessible from some kind of separate body of content, like a database (in a wide sense of the word).

Ontologies and Semantic Networks

2.1 Ontologies and Semantic Networks

One way of representing semantic knowledge is through ontologies. In its original meaning, ontology is a branch of Philosophy concerned with the study of the nature and relations of things that exist. However, in Computer Science in general—and in Information Systems and Artificial Intelligence (AI) in particular—the word ontology has two related but slightly different meanings; the first refers to a representation vocabulary, while the second refers to a body of knowledge, often described using a representation vocabulary (Chandrasekaran et al. 1999).

Sowa (2000; p. 492) gives a definition of ontologies that reflects their use in Computer Science:

> The subject of *ontology* is the study of the *categories* of things that exist or may exist in some domain. The product of such a study, called *an ontology*, is a catalog of the types of things that are assumed to exist in a domain of interest $\mathcal{D}$ from the perspective of a person who uses a language $\mathcal{L}$ for the purpose of talking about $\mathcal{D}$. The types in the ontology represent the *predicates*, *word senses*, or *concept and relation types* of the language $\mathcal{L}$ when used to discuss topics in the domain $\mathcal{D}$.

Hence, people in Computer Science often talk about *an ontology* as an engineering artifact for a particular domain. Examples of such domains of interest are: bus travel (Amble 2000), Ornithology (Jönsson et al. 2004), and Molecular Biology (Ashburner et al. 2000; Sætre 2006).

The same semantic resources are often referred to as both ontologies and semantic networks. This might seem a bit imprecise, so a short clarification would be in order. According to Russel and Norvig (2003; p. 349) semantic networks "provide graphical aids for visualizing a knowledge base and efficient algorithms for inferring properties of an object on the basis of its category membership." Hence, both *properties of objects* and *categories*—or classes—are described in both ontologies and semantic networks. Furthermore, Sowa (2000; p. 495) notes that a knowledge base (KB) is "an informal term for a collection of information that *includes* an ontology as one component" (emphasis added by me).

2.1.1 *Example Ontologies*

Some examples of large-scale general ontologies are WordNet[2] (Fellbaum 1998c), CYC (Lenat 1995), and OMEGA (Philpot et al. 2003). OMEGA is the successor of SENSUS (Knight and Luk 1994), which will be briefly presented in Section 2.4.

Since WordNet constitutes a central part of the research presented in this dissertation, it will also receive the most attention below. Different lexical semantic concepts and the WordNet ontology will be presented in more detail in sections 2.2 and 4.1.2, respectively.

However, to give an impression of what a real-world ontology looks like, Figure 2.1 on the following page shows an extremely small portion of the WordNet ontology. Each concept in the figure is represented as a set of synonymous senses of English words. Each sense is written in the format

```
<word>_<lexical category>_<sense number>,        (2.1)
```

where `<word>` identifies the word, `<lexical category>` signifies the lexical category of the sense. In WordNet, the lexical category can be either noun (`n`), verb (`v`), adjective (`adj`), or adverb (`adv`). Finally, the `<sense number>` is a number that is used to distinguish between other senses of the same word in the same lexical category.

The only relations represented in Figure 2.1 are the hyponymy—or is-a-kind-of—relation (if one follows the edges from bottom to top) and the synonymy relation. Thus, the figure shows that a `cousin_n_1` *is a kind of* `relative_n_1`, and—through the transitivity of the hyponymy relation—is ultimately a kind of `entity_n_1`. This way, the hyponymy relation provides a partial ordering and classification of concepts. Figure 2.1 also shows that, for example, the senses `person_n_1` and `individual_n_1` are synonymous.

2.1.2 *Applications of Ontologies in Natural Language Processing*

The knowledge provided by ontologies can be used by applications in several sub-fields of Natural Language Processing (NLP). Some of the NLP applications where ontologies are required—or at least can be put to great use—will be presented next.

2 Sometimes referred to as Princeton WordNet to differentiate between several kinds of wordnets. However, herein WordNet will by default refer to the Princeton version.

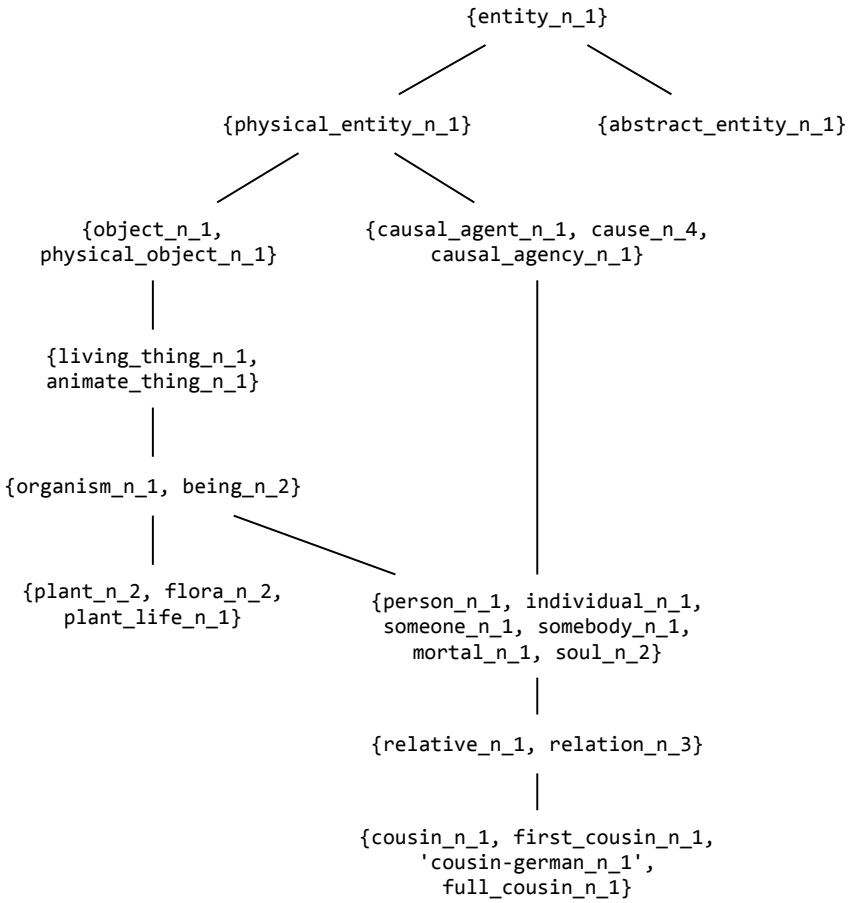


FIGURE 2.1: Part of WordNet's taxonomy. Each edge in the graph designates a generalization/specialization relationship between two concepts.

Word-Sense Disambiguation

Word-Sense Disambiguation (WSD), or lexical disambiguation, is the process of deciding which of several different senses of a word is intended in a given context. The task of disambiguating word senses is not a goal in itself, but rather an important part of several other NLP tasks, including parsing, semantic analysis, Information Retrieval (IR), knowledge acquisition, and Machine Translation (MT).

Resnik (1998) and Stevenson and Wilks (2001) show how semantic knowledge sources can be used in WSD. For example, the nominal "seal" is ambiguous between—at least—"aquatic carnivorous mammal" and "seal of approval". However, give the sentence

> "The seal chased a shoal of fish."

a WSD system can use semantic information to see that the verb "chase" specifies a selectional restriction[3] that its subject should be animate.

Moldovan and Novischi (2004) show how they use automatic semantic disambiguation of the words in the WordNet glosses to build their eXtended WordNet from the WordNet 2.0 lexical semantic network.

Text Summarization

Text Summarization is "the process of distilling the most important information from a source (or sources) to produce an abridged version for a particular user (or users) or task (or tasks)" (Mani and Maybury 1999). The goal of the text summarization process is typically to produce summaries that "convey maximal information in minimal space" (McKeown et al. 1995).

Not all approaches to automatic text summarization make significant use of lexical-semantic knowledge as found in NLP ontologies. For example, Extractive Text Summarization aims only to extract the most noticeable sentences of a text. However, when doing deeper characterizations of texts, like in Abstractive Text Summarization, for example knowledge about synonymous senses of words can be used to detect that multiple passages of text discuss the same concept with different words.

Document Clustering

Document Clustering is the process of—often hierarchically—grouping together documents that are similar according to a set of predefined

3 Trask (1993) defines selectional restriction as "any of various semantic constraints reflected in the ability of lexical items to combine in syntactic structures."

characteristic features. Document clustering can be utilized in several different areas. For example, document clustering can be used for improving IR systems by increasing the relevancy of the returned set of documents (Salton et al. 1975), or large-scale topic discovery in large text collections (Larsen and Aone 1999; Ayad and Kamel 2002).

A common way of clustering documents involves two phases. In the first phase each document is mapped to a point in a high-dimensional space based on a vector representation of the document's characteristic features. During the second phase, algorithms automatically organize the documents into a hierarchy of clusters. However, the vectors used in this document clustering approach constitute a bag-of-words representation (Joachims 1997) of the documents that ignores, for example, conceptual similarity that exist between different terms that does not literally co-occur. Hotho et al. (2003) show examples of how the integration of an ontology improves text clustering results by replacing the terms in the bag-of-words vector with semantic concepts found in the ontology. This makes information about, for example, synonymy and hyponymy available for exploitation by their clustering algorithms.

Information Extraction

Information Extraction (IE) is "the automatic identification of selected types of entities, relations, or events in free text" (Grishman 2003).

As an example of how lexical semantic information may be used by an IE system, assume that the system is processing the text fragment

> "... Diamond sang 'America'... "

Given this input, the IE system must decide whether to classify the noun phrase "diamond" as a thing or a person. If such a system has access to an ontology with semantic knowledge about verb frames, it can rule out the "thing" classification because the noun phrase is the subject of the communication verb "sing", which prefers an animate subject.

Information Retrieval

IR, which is also known as document retrieval or text retrieval, is the process of "finding material (usually documents) of an unstructured nature (usually text) that satisfies an information need from within large collections (usually stored on computers)" (Manning et al. 2008). Since the advent of document retrieval for the World-Wide Web (WWW), IR systems are often called search engines.

A modern IR system often utilizes both WSD (Voorhees 1993) and Document Clustering—mentioned above—in addition to several other task-specific techniques. Therefore, we have already seen how one may use an ontology to improve the results produced by an IR system.

However, there are additional ways that a search engine can utilize an ontology. For example, both Voorhees (1998) and Moldovan and Mihalcea (2000) show how query expansion can improve the performance[4] of a search engine. Often, the queries a search engine receives from users consist of one to three words. At the same time, a concept relevant to the user might be referred to with different but synonymous words in different documents. Because of this, the search engine might fail to retrieve many of the relevant documents because the user does not enter all the relevant synonymous words. To remedy this, query expansion attempts to increase the number of relevant documents in the result set by adding semantically similar senses—found in the ontology—to the user's query.

Gonzalo et al. (1998) and Verdejo et al. (2000) show how the multilingual EuroWordNet (Vossen 1998; Peters et al. 1998; Vossen 2004) can be used to perform cross-language IR for the languages English, Spanish, and Catalan. An interlingual index that connects each of the monolingual wordnets for English, Spanish, and Catalan, that are part of EuroWordNet, makes it possible to find concepts that are equivalent to those used in the query in complement languages.

Text Interpretation

Text Interpretation, also known as semantic interpretation and NLU, is the process of extracting and representing the meaning of textual natural language input (Hobbs et al. 1993). As stated by Jacobs and Rau (1993) "the end product of text interpretation is to produce some clearly prescribed structures from the input, with measurable accuracy."

The resulting meaning-representation structures are used in many NLP tasks, including knowledge acquisition (Gomez 1994), question-answering (QA) (Gomez et al. 1994; Hull and Gomez 1999; Amble 2000) and Machine Translation (MT) (Knight and Luk 1994; Mahesh and Nirenburg 1996). In such systems, the ontology may be used both during semantic analysis of the textual input and as a central resource for reasoning about the knowledge derived from the textual input.

4 As noted by Moldovan and Mihalcea (2000), "performance is a standard information-retrieval system measure of the number of relevant documents retrieved over the total number of documents retrieved." This measure is often also called the precision of a method or system.

One of the major differences between IE and knowledge acquisition is that the result of an IE process is typically geared towards ultimately being interpreted by a human, often with an intermediary IR system, while the result of a knowledge acquisition process is typically used by a computer system able of reasoning. Therefore, IE systems can apply lighter-weight methods to extract relevant information, while knowledge acquisition systems typically perform more resource-demanding, deeper semantic analyses. Hence, IE systems are able to process much larger volumes of text, consuming less resources (manifested in higher speed), than knowledge acquisition systems.

Multilingual ontologies, where lexicon entries in each language are mapped to concepts in the ontology, are important resources in many MT systems because they are used in common language-independent representations of meaning.

Natural Language Generation

Natural Language Generation (NLG) is "concerned with the construction of computer systems that can produce understandable texts in English or other human languages from some underlying non-linguistic representation of information" (Reiter and Dale 1997).

NLG is typically needed to make information represented in ways well suited for structured storage and manipulation by computer systems easy for humans to comprehend. Examples of this can be transformation of information from tabular material, relational databases, or logical formulas into natural language. For example, in an article by Nordgård et al. (2005) we presented an approach to automatically producing natural language sentences from a set of unstructured semantic concepts.

Jing and McKeown (1998) show how semantic knowledge bases can be used in NLG by using WordNet both for selecting appropriate lexical representations of the semantic input and for choosing the words in the final surface realization, thereby using the synonymy relation in WordNet to generate lexical paraphrases.

2.2 Lexical Semantics

This section provides a preliminary introduction to the theory of lexical semantics with focus on topics that are central to the method for automatically mapping Norwegian content words[5] to WordNet senses that will be presented in Chapter 4.

5 The term *content words* will be explained below.

The word *fly* means different things in different contexts. For example, in the sentences "Fly me to the moon" and "There is a fly in my soup" the word *fly* has two very different meanings. The first (verb) meaning is of an action, while the second (noun) describes an insect. The two meanings of the word represent different lexemes. A lexeme—also called a lexical item—is an abstract entity that "has a more-or-less consistent meaning or function but which can vary in form for grammatical purposes" (Trask 1993). The different forms of a lexeme are called word forms. Another name for word form is morphosyntactic word.

For example, the first lexeme above can be expressed by the set of word forms {*fly*, *flies*, *flew*, *flown*, *flying*} that represent different temporal information about the verb, while the second can be expressed by {*fly*, *fly's*, *flies*, *flies'*} which represent the singular and the plural form of the noun. When two lexemes share the same word forms like this, they are called homographs. Homographs may also share the same lexical category. A word's lexical category is also often referred to as its word class or POS.

A base form of a lexeme is usually the form where no inflection has been applied. For example, both *smiling* and *smiled* are inflected forms of the base form *smile*. Most dictionaries are organized as alphabetized lists where the base form of each lexeme it describes is used as a keyword.

Words belonging to the lexical categories nouns, verbs, adjectives, and adverbs are often called *content words*, in contrast to function words or grammatical words. Probably the most important difference between content words and function words is that the meaning of content words can be found in a dictionary[6], while a function word carries little meaning in itself, so a dictionary usually only describes its usage. The function words belong to lexical categories such as prepositions, interjections, auxiliary verbs, and determiners.

The word classes nouns, verbs, adjectives, and adverbs constitute open word classes. The other word classes represent closed word classes. In contrast to an open word class, a closed word class contains relatively few words, and new words are very seldom added to it.

Since the method presented herein is based on mapping of content words, the following sections will not focus on function words. In the next sections, relevant semantic relations will be introduced. Because Verto uses WordNet as its semantical resource, the presentation of the semantic relations will be linked to how they are defined in WordNet.

6 That is, if the dictionary contains definitions.

2.2.1 *Synonymy*

Propositional logic is concerned with the formal properties of statements that are either *true* or *false*, called propositions (Gamut 1991b; pp. 28–64). Generally, such statements are represented by grammatical declarative sentences.

Cruse (1986; p. 88) gives the following definition of synonymy from a propositional perspective:

> X is a propositional synonym of Y if (i) X and Y are syntactically identical, and (ii) any grammatical declarative sentence S containing X has equivalent truth-conditions to another sentence S', which is identical to S except that X is replaced by Y.

For example, if we let S and S' denote the sentences "The ring he gave her was made of *24-karat gold*" and "The ring he gave her was made of *pure gold*" respectively, we see that *24-karat gold* and *pure gold* are propositional synonyms, according to the definition above.

However, lexemes that are synonyms in one context need not be synonymous in another context. As Hasselgård et al. (1998; p. 54) write, "[...] it is very difficult to find two lexemes which are completely synonymous. The test would be if they can be used interchangeably in *all* contexts" (emphasis added by me). Such synonymy, called absolute synonymy, occurs very seldom in the real world. This notion is also expressed by Miller (1998; p. 24), "[...] by that criterion, natural languages have few synonyms."

Therefore, the criterion for lexemes to be synonymous often adopted in practice is that synonyms must be interchangeable—without changing the truth-value—in *some* contexts (Saint-Dizier and Viegas 1995; p. 18). This definition of synonymy is also used in WordNet (Miller 1998; p. 24).

2.2.2 *Hypernymy and Hyponymy*

If a lexeme X denotes a superordinate of Y (that is, if X belongs to a more general category than Y), then X is a hypernym of Y. In the same situation, Y is said to be a hyponym of X. As mentioned in Section 2.1.1, the hyponymy relation is also called the is-a-kind-of relation, because if Y is a hyponym of X, then Y is a kind of X. Both hyponymy and hypernymy are transitive relations. That means if Z is a kind of Y and Y is a kind of X, then Z is a kind of X, and the same holds for hypernymy.

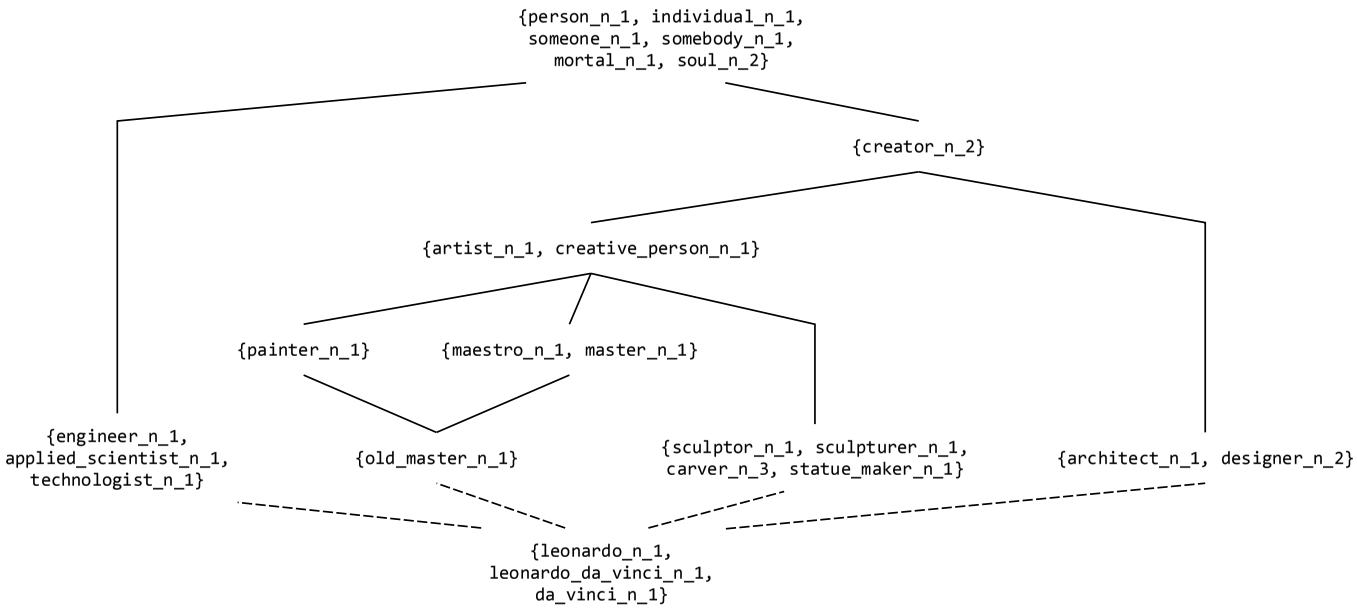


FIGURE 2.2: Hypernymy/hyponymy in the WordNet ontology. The stippled lines represent instance relations, while the solid ones represent class relations.

Hypernymy defines the level of generality of a concept relative to other concepts. For example, *mammal* is more general than *whale*, so one can safely state that "all whales are mammals", but the statement "all mammals are whales" is untrue.

It is possible to distinguish between two kinds of hypernymy and hyponymy. The first—general—kind is said to be a relation between classes (or types) of things, while the second—sometimes called instance hypernymy and instance hyponymy—is a relationship between an instance and a class. When this distinction is made, the instance hyponymy relation is also called is-a. For example, in Figure 2.2, *Leonardo da Vinci is a* sculptor, and he *is an* engineer, because da Vinci is an *instance* of the *classes* of engineers and sculptors. However, an engineer *is a kind of* person (note that the stippled lines represent instance relations).

The complete network structure defined by the hypernymy and hyponymy relation constitutes a taxonomy. Since a concept in WordNet can have multiple hypernyms—see for example the concept `old_master_n_1` in Figure 2.2—the full taxonomy of WordNet constitutes a lattice or heterarchy[7] structure.

2.2.3 *Troponymy*

One can say that troponymy is to verbs what hyponymy is to nouns. That is, troponyms more precisely describe in what manner a more general action is performed. As written by Fellbaum (1998b; p. 79), the "troponym relation between two verbs can be expressed by the formula *To* V_1 *is to* V_2 *in some particular manner*", where V_1 and V_2 denotes the verbs. For example, "To *smile* is to *grimace* in some particular manner."

2.2.4 *Antonymy and Similarity*

Antonymy is a semantic relation between two adjectival lexemes and denotes that the lexemes represent opposite meanings. Moreover, just as for the synonymy relation, because this requirement seldom is satisfied in *all* contexts, two lexemes X and Y are considered antonyms if they express opposite meanings in *some* contexts.

Furthermore, it should be noted that, as Hasselgård et al. (1998; p. 56) write, "antonymy is a tricky notion." For example, if asked what the opposite of *close* is, most would probably answer *distant*, and not *far* even though *near* and *close* in some contexts are synonyms (as in

7 In a hierarchy, each concept in the taxonomy has at most one parent, and zero or more children. A heterarchy is similar, except that each node may have multiple parents; a situation that corresponds to multiple inheritance in object-oriented (OO) modeling.

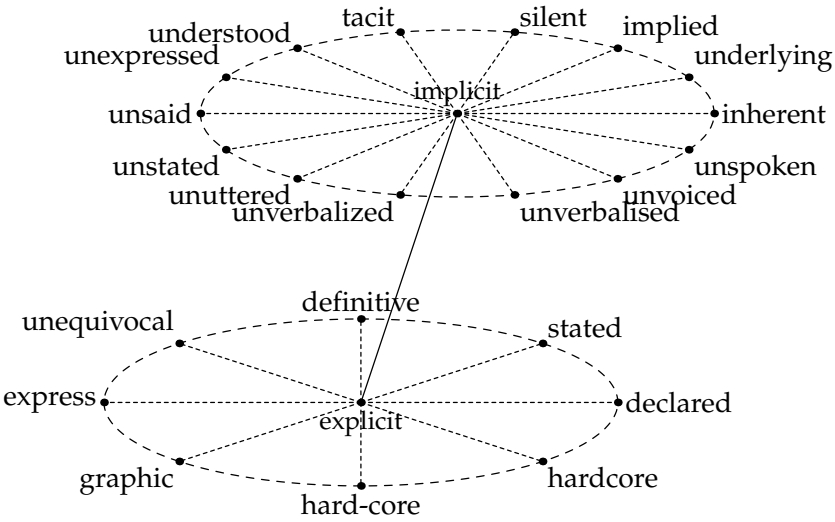


FIGURE 2.3: Similarity between adjectives and direct and indirect antonymy. The two direct antonyms are related to each other like opposed hubs connected by an axle, while the indirect antonyms are related to each hub like the spokes of a wheel. All the adjectives along the rim of each wheel are similar in meaning to the adjective at the hub.

"getting close to something"). Somehow, some pairs of adjectives just seem to belong together as antonyms; like *close*/*distant* and *near*/*far*. Such pairs are called direct antonyms.

On the other hand, some adjectives, like *tacit*, *underlying*, and *inherent*, do not seem to have any direct antonyms. Nevertheless, all of these three adjectives share a similar meaning, namely *implicit*. Likewise, *definitive*, *graphic*, and *stated* are all similar in meaning to *explicit*. Furthermore, *implicit* and *explicit* form a direct antonym pair. Hence, the adjectives that are similar to *implicit* and the adjectives that are similar to *explicit* are said to be indirect antonyms. This way of classifying direct and indirect antonyms is used to define the similarity relation between adjectives in WordNet (Fellbaum 1998b; pp. 48–52). Figure 2.3 shows how these concepts relate to each other in a concrete example.

2.2.5 *Compounds and Collocations*

A compound is a lexeme that consists of multiple lexemes, whereas a simplex is a simple word consisting of a single lexeme.

Compounds can be either endocentric or exocentric, but most compounds are endocentric. An endocentric compound consists of a grammatical head and a dependent. The head carries the central meaning of the compound and the dependent modifies the meaning of the head. The head will also determine the syntactic category for the whole compound.

Furthermore, as stated by Trask (1993; p. 91), an endocentric compound is "a hyponym of the grammatical head." For example, *steamboat* is a hyponym of *boat*, where *boat* serves as the head of the lexeme, while *steam* functions as its modifier. In English, the head of a compound is always the rightmost lexeme.

Compounds need not be nouns. For example, *nosedive* is both a verb and an endocentric compound.

The term exocentric compound[8] refers to a compound where the meaning and syntactic category of the whole do not have any direct connection to the grammatical head. Hence, an exocentric compound is not composed of a distinct head and dependent. An example exocentric compound is *bluebell (Scilla non-scripta)*, a kind of plant; *not* a kind of neither blue nor bell. Another example would be *buttercup (Ranunculus acris)*, another kind of plant.

2.3 Automatic Analysis of Norwegian Compounds

A lexicon or dictionary usually covers only a limited set of words of a language. Usage frequencies often decide which words are covered. No matter how many words are covered, the lexicon cannot cover words unknown at the time the lexicon was constructed.

However, in Norwegian new words are formed all the time, and as stated by Johannessen (2001), "compounding is probably the most important and widely used word formation process in Norwegian." This means that from the moment of publication of a lexicon there is a risk that it does not cover some new word—probably formed through compounding. On the other hand, if the dictionary does not cover a compound, it most probably contains the constituents of the compound. Therefore, determining the meaning of the compound is usually a matter of determining the meaning of each of its constituents.

8 Exocentric compounds are also referred to as *bahuvrihi* compounds.

There is surprisingly little research published on the topic of automatic Norwegian compound-word analysis. The only notable research is presented by Johannessen and Hauglin (1996) and Johannessen (2001). Their main thesis is that the constituents of most Norwegian compounds are stems, and not independent word forms, sometimes joined together by -s- or -e- epentheses. These epentheses are sometimes introduced to ease—or to reflect the actual—pronunciation of a compound. For example, the -s- in «*aluminiumsklump*» ("aluminum lump") and the -e- in «*barnemat*» ("child's food/child's play") are both epenthetic. However, correctly identifying the constituent parts of a compound through automatic analysis, sometimes referred to as compound segmentation, is not a trivial task, and epentheses often introduce ambiguities that are difficult to disambiguate.

It should be noted that even though little research on Norwegian compounds has been published, the problem is of practical interest to many kinds of software that process Norwegian text. For example, practically all everyday, off-the-shelf software for word processing that support spelling correction for Norwegian contain solutions to this problem.

Furthermore, since the Scandinavian languages Danish, Norwegian, and Swedish, are similar enough that most Scandinavians are able to understand each other correctly most of the time, research on automatic compound-word analysis for Danish and Swedish is also relevant.

For example, Karlsson (1992) presented SWETWOL, an implementation of a tool for morphological analysis of Swedish text, with an emphasis on computational analysis of Swedish productive compounds. SWETWOL uses morphosyntactic information and no semantic information during its analyses.

Kokkinakis et al. (2000), reported that they use a heuristic for their compound segmentation algorithm based on 3-grams and 4-grams that are *complementary* to a set of 3-grams and 4-grams that represent character sequences that occur in non-compound words. Their segmentation process involves applying the complementary 3-grams and 4-grams to identify segmentation locations in compounds through finding character sequences that are disallowed in non-compound words. Kokkinakis et al.'s compound-word analyzer also uses some semantic information from the Swedish SIMPLE lexicon (Lenci et al. 2000).

Furthermore, Pedersen (2007) presents research on improving IR search results through combining query expansion through splitting of compounds with shallow parsing to remove irrelevant results. Unfortunately, she provides no presentation of the algorithms used by the

compound-word analyzer. However, the performance of the analyzer will be commented in Section 3.7.

To handle compound words, Verto—the implementation of the method to be presented in Chapter 4—incorporates a module for automatic analysis of Norwegian compounds based on morphological structures of words. The module's grammar and algorithms are based on the linguistic principles presented by Johannessen and Hauglin (1996) and Johannessen (2001), with some adjustments and additions.

2.4 Lexical Semantic Resources for Non-English Languages

SENSUS, mentioned above, is a large-scale ontology designed to be used in MT and was created by semi-automatically merging several lexical and semantic resources, including WordNet, the *Longman Dictionary of Contemporary English* (LDOCE) (Procter 1978), the Penman Upper Model (Bateman 1990) and *Collins English–Spanish/Spanish–English Dictionary* (Collins and Smith 1971). Part of the merge process consisted of semi-automatically mapping Spanish words to English concepts in the ontology (Knight 1993; Knight and Luk 1994). Their methods for mapping Spanish words to English concepts will be described in Section 2.4.3.

Obviously, Knight and Luk's approach relates to mine because they create a mapping from words in a non-English language, Spanish, to English "ontological entities" in an ontology built by merging WordNet and LDOCE. However, their approach does not make use of the semantic relations in WordNet in the same way, and they do not perform *inverse translations*[9], a concept that is crucial to the method I present herein. The importance of inverse translations to my method will be described in Section 4.5.

Okumura and Hovy (1994) present work on mapping Japanese words to English, using the same algorithm as described by Knight and Luk (1994).

EuroWordNet (Vossen 1998; Alonge et al. 1998) is a multilingual database containing separate semantic networks, referred to as wordnets, for several of the European languages. The development was done as a European Union (EU) project, and the first languages included were Dutch, Spanish, Italian, English, French, German, Czech, and Estonian. A separate wordnet was developed for each language, and each wordnet was then connected to an inter-lingual index by identifying

9 Kevin Knight, e-mail message to Martin Thorsen Ranang, July 13, 2006.

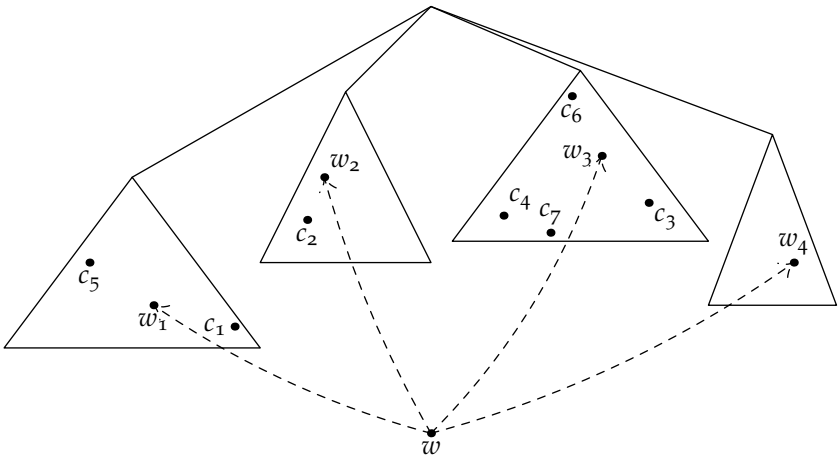


FIGURE 2.4: Illustration of the conceptual density measure by Agirre and Rigau (1996).

equivalent concepts in each wordnet. Later, the same principles that were used in EuroWordNet have been used to create wordnets for other languages, such as the Balkan languages (Tufiş et al. 2004).

Semantic Information for Multifunctional Plurilingual Lexica (SIMPLE) (Lenci et al. 2000) is a semantic resource that has already been developed for twelve of the European languages; namely, Catalan, Danish, Dutch, English, Finnish, French, German, Greek, Italian, Portuguese, Spanish, and Swedish. The SIMPLE project has added semantic information to the already present morphological and syntactic description for a subset of roughly 6,000 words, or approximately 10,000 lemmas, in the *Preparatory Action for Linguistic Resources Organisation for Language Engineering* (PAROLE)-lexicon. The SIMPLE approach is different from EuroWordNet in that is based on the Generative Lexicon theory of Pustejovsky (2001), and the semantic information is represented through a kind of extended qualia structures. The semantic information is specified partly as semantic relations and partly as semantic features. Hence, SIMPLE is not developed mainly as a semantic network. The network structure that emerges can be seen as a side effect of adding semantic descriptions for each word.

2.4.1 Conceptual Density

Rigau and Agirre (1995) present two methods they have explored for mapping both French and Spanish nominals to WordNet 1.5. In the first method they present, they combine the French–English and the English–French parts of a French–English bilingual dictionary to increase the number of available translations. They then map French nouns to WordNet while, in the case of polysemous English nouns, they try to infer the correct sense of the translation based on a measure they call semantic density, or conceptual density (described in depth by Agirre and Rigau 1996).

Conceptual density can be used as a measure of relatedness between words. Rigau and Agirre divide WordNet into subhierarchies, based on hypernymy/hyponymy relations. The conceptual density of a word sense and a given context of words is defined by the number of context words that occur within the same WordNet subhierarchy as the sense in question, relative to the size of the subhierarchy. The size of the subhierarchy is defined as the total number of senses it contains. The context words include dictionary cue words—translated from French using the bilingual dictionary—and words surrounding the noun in question if it stems from a multiword translation. Given a translation from French to an English noun with multiple senses found in WordNet, Rigau and Agirre's first method chooses the sense belonging to the subhierarchy with the highest conceptual density.

For example, Figure 2.4 shows a word, w, with senses w_1, w_2, w_3, w_4, each belonging to a separate subhierarchy of WordNet, and a set of context words, $c_1, c_2, \ldots, c_7$. Because the subhierarchy that contains w_3 has the most such occurrences, relative to the size of the subhierarchy is, Rigau and Agirre's first method would return w_3 for w. If multiple senses of a word had belonged to the same subhierarchy, all senses would be returned.

For testing their second method, Rigau and Agirre (1995) repeated the act of merging the two halves of a bilingual dictionary, but this time it was a Spanish–English/English–Spanish dictionary. However, their second method is less interesting, because it simply accepts all possible translations, or mappings, without any attempt at disambiguating the mapping. Thus, if a polysemous English word has several translations into Spanish, their method would create mappings from each of the Spanish words to each of the WordNet senses.

2.4.2 *Conceptual Distance*

A measure closely related to conceptual density is the conceptual distance measure, described by Agirre et al. (1994). The conceptual distance between two words, w_1 and w_2, is defined as

$$\text{dist}(w_1, w_2) = \min_{\substack{c_{1_i} \in w_1 \\ c_{2_j} \in w_2}} \sum_{c_k \in \text{path}(c_{1_i}, c_{2_j})} \frac{1}{\text{depth}(c_k)}, \tag{2.2}$$

where c_{1_i} and c_{2_j} are WordNet synonym sets (synsets), or concepts, representing w_1 and w_2, respectively, and $\text{depth}(c_k)$ defines a synonym set (synset)'s specificity according to its depth in the WordNet hierarchy. In other words, the conceptual distance between w_1 and w_2 is defined through the specificity of synsets, c_k, along the shortest path that connects c_{1_i} and c_{2_j}. Concepts deeper in the hierarchy are considered more specific, and more specific concepts are considered more closely related to each other.

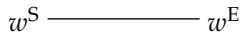
By applying the formula to a pair of words, one can find the most closely related synset pair that represents those words. Just like conceptual density, the conceptual distance closeness measure between related senses can be used to disambiguate mappings of non-English words to WordNet. For example, one can use some kind of context words (as described in the previous section) and select the sense that has the shortest conceptual distance to senses of the context words (see, for example Rigau 1994; Atserias et al. 1997).

2.4.3 *Other Methods*

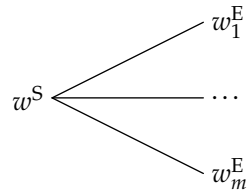
Atserias et al. (1997) present criteria and methods they explored for automatically creating a multilingual lexical knowledge base, or lexical-semantic resource, using a Spanish–English/English–Spanish bilingual dictionary (Vox-Harrap 1992) and a large Spanish monolingual dictionary (Alvar Ezquerra 1987) for mapping Spanish *nouns* to WordNet 1.5. They do not state whether their methods are able to handle nonlexicalized Spanish compounds, or compounds that are not present in the dictionary resources.

Atserias et al.'s work was included in the EuroWordNet project to help (semi-)automatically build the Spanish part of EuroWordNet, known as Spanish WordNet. Even though Atserias et al.'s research is concerned with mapping Spanish nouns to WordNet, their criteria and methods could be considered independently of source language.

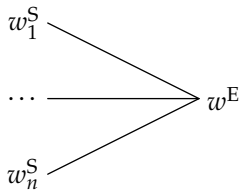
Their research provides a wide spectrum of criteria and methods for using bilingual dictionary resources mapping non-English words to



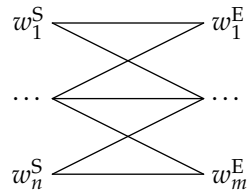
(a) Only one possible translation, to a single-sense English word, w^E. The Spanish word, w^S, gets linked to the synset containing the corresponding sense.



(b) One Spanish word, w^S, with several possible translations to single-sense English words, $w_1^E, w_2^E, \ldots, w_m^E$. The Spanish word gets linked to the synsets containing each corresponding sense.

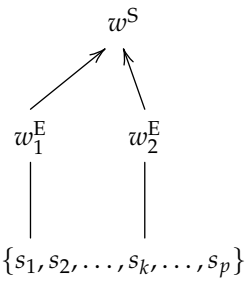


(c) One monosemous English word, w^E, is the translation target of several Spanish words. Each of the Spanish words, $w_1^S, w_2^S, \ldots, w_n^S$, gets linked to the unique synset.

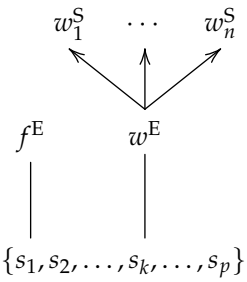


(d) More than one Spanish word, $w_1^S, w_2^S, \ldots, w_n^S$, have the same single-sense English words, $w_1^E, w_2^E, \ldots, w_m^E$, as translation targets. Each Spanish gets linked to each corresponding target synset.

FIGURE 2.5: Situations and criteria for mapping source-language words to WordNet synsets via *monosemous* English words. Based on criteria described by Atserias et al. (1997).



(a) Variant criterion. More than one sense of a given synset—in this case s_1 and s_k, representing the English words, w_1^E and w_2^E, respectively—have only one translation each, and those translations target the same Spanish word, w^S. The Spanish word gets linked to the synset in question.



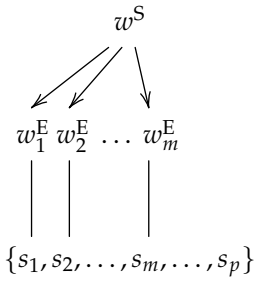
(b) Field identifier criterion. Utilizing field identifier information provided by the English–Spanish dictionary. If senses representing both the English word to be translated, w^E, and the field identifier, f^E, occur in the same synset, then a link is created for each translation of w^E into Spanish, $w_1^S, \ldots, w_n^S$ found in the dictionary.

FIGURE 2.6: Methods and criteria for mapping source-language words to WordNet synsets via *polysemous* English words. The figures visualize the criteria described textually by Atserias et al. (1997).

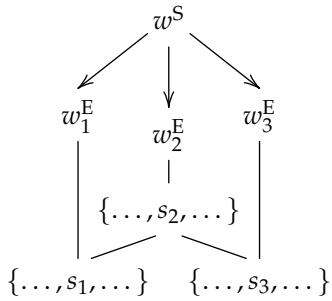
WordNet synsets. Therefore, in figures 2.5 to 2.7 on pages 28–30 I have illustrated the criteria and methods textually described by Atserias et al. to more easily convey the ideas of their approach.

Figure 2.5 on the facing page shows how mappings are established when the Spanish–English bilingual dictionary provides translations to monosemous, or single-sense, English words. Generally, mappings are established from Spanish words to the synset that contains the unique sense representing each monosemous English word involved in a translation. Mapping of non-English words to WordNet synsets that represent monosemous English words constitutes the simplest case of mapping non-English words to WordNet.

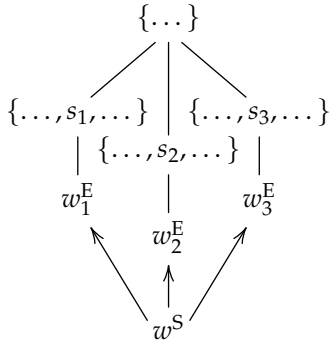
If the English words have several possible meanings, deciding which mappings are warranted becomes less trivial. Figure 2.6 shows methods and criteria for mapping Spanish words to WordNet synsets when the English translations suggested by the bilingual dictionaries are polysemous. Because Figure 2.6a, called the "variant criterion" by Atserias et al., could resemble some aspects of my method, it will be analyzed and discussed in Section 4.11.5.



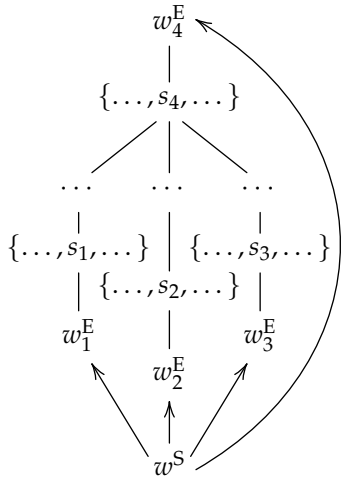
(a) Intersection criterion. Given a translation from a Spanish word, w^S, to several possible polysemous English words, $w_1^E, w_2^E, \ldots, w_m^E$, a mapping from the Spanish word is created if senses of all the English words share at least one synset.



(b) Direct hypernym criterion. Given a translation from a Spanish word, , w^S, to multiple polysemous English words, for example w_1^E, w_2^E, w_3^E, if a sense representing one of the words is a direct hypernym of senses representing the other words, a link is created from the Spanish word to all the hyponym synsets.



(c) Sibling criterion. Given a translation from a Spanish word, w^S, to multiple polysemous English words, for example w_1^E, w_2^E, w_3^E, if senses representing each of the words are *siblings*, sharing some other hypernym synset, a link is created from the Spanish word to all the hyponym synsets.



(d) Distant hypernym criterion. Equal conditions as in Figure 2.7b, except that the common hypernym, in this case w_4^E, is a distant, *not direct* hypernym.

FIGURE 2.7: Methods and criteria for mapping source-language words to WordNet via polysemous English words, using WordNet's structural information. The figures visualize the criteria described textually by Atserias et al. (1997).

Figure 2.7 on the preceding page shows methods and criteria that use more of WordNet's structural information to decide which mappings are correct when the translations contain polysemous English words. In addition to the methods and criteria described by figures 2.5 to 2.7, Atserias et al. use three methods based on the conceptual distance measure.

Farreres et al. (1998) present work that extends the approach described by Atserias et al. (1997) to create Spanish WordNet, part of the EuroWordNet initiative, and Catalan WordNet. They first apply the methods described by Atserias et al. to map Spanish and Catalan nouns and verbs to WordNet 1.5. Then, Farreres et al. present a method they have applied to extend each of those resources by merging them with accurate taxonomies generated from Spanish and Catalan monolingual dictionaries.

Farreres et al. (2002) use different statistical analyses to evaluate the quality and accuracy of the methods presented by Atserias et al. (1997). By also studying the—sometimes overlapping—contribution of mappings from each method, Farreres et al. are able to generate detailed coverage and accuracy statistics, trading coverage for increased accuracy and vice versa, for Atserias et al.'s approach.

The methods Knight and Luk (1994) use to map Spanish words to their WordNet-based ontology are variants of the criteria shown in figures 2.5a (a single unambiguous English translation), 2.6a (the variant criterion), 2.7c (the sibling criterion), and 2.7d (the distant hypernym criterion). When their Spanish–English bilingual dictionary provides translations containing multiple English words, they first try the variant criterion. If the variant criterion cannot be matched, they first try the sibling criterion and after that the distant hypernym criterion. Knight and Luk's distant hypernym criterion is really a blend of the sibling and the distant hypernym criteria, because the value of the common hypernym, w_4^{E}, in Figure 2.7d does not matter. However, the farther away, measured by the number of edges in the graph, that the common hypernym is, the lower confidence level the mapping has. Because Knight and Luk's ontology was built by merging LDOCE with WordNet, they are able to use field codes (such as COM and ZOOL for meanings related to commerce and zoology, respectively) provided by the bilingual dictionary (Collins and Smith 1971). They use these field codes for guiding the disambiguation of mappings when only one English word is given as a translation of a Spanish word.

Kokkinakis et al. (2000) present methods for (semi)automatically extending and enriching the coverage of the Swedish SIMPLE lexicon. They utilize semantic information available from the Swedish SIMPLE

lexicon[10], the *Gothenburg Lexical Data Base* (GLDB)[11], and large corpora. Their first method is based on first analyzing compound words not already present in the Swedish SIMPLE lexicon to find their constituent parts. They then incorporate the compounds in the lexicon with information about semantic type, domain, and semantic class inherited from the head of the compound. Their second method is based on a hypothesis that when several nouns are listed in an enumerative fashion, if all the *known* elements are of the same semantic class and co-hyponyms, or siblings as shown in Figure 2.7c, then the *unknown* elements of the enumerative noun phrase are probably also siblings of the same parent hypernym. Thus, by the transitivity of the hyponymy relation, the new words can be incorporated into the lexicon while inheriting semantic information from its hypernym.

Dorr (1997) describes techniques for constructing large-scale dictionaries for foreign-language tutoring and interlingual MT systems. Parts of her work is focused on automatically building dictionaries based on a structured lexical, language-independent, or interlingual, representation of verb meaning called lexical conceptual structure (LCS) (Dorr 1992, 1993). The LCS approach is based on Jackendoff's (1992) theory on semantic structures.

LCS captures lexical knowledge related to argument structures, thematic roles, and selectional restrictions. However, LCS does not capture deeper kinds of semantics, such as domain and world knowledge. For example, the LCS

$$
\begin{array}{l}
[_{\text{Event}}\ \text{GO}_{\text{Loc}} \\
\quad ([_{\text{Thing}}\ \text{JOHN}], \\
\quad\ [_{\text{Path}}\ \text{TO}_{\text{Loc}} \\
\quad\quad ([_{\text{Thing}}\ \text{JOHN}], \\
\quad\quad\ [_{\text{Position}}\ \text{AT}_{\text{Loc}}\ ([_{\text{Thing}}\ \text{JOHN}], [_{\text{Thing}}\ \text{SCHOOL}])])], \\
\quad [_{\text{Manner}}\ \text{JOGGINGLY}])],
\end{array}
\tag{2.3}
$$

from Dorr (1997), represents the sentence "John jogged to school."

The manual, interactive, and corpus-based techniques Dorr (1997) presents are of little interest to the work presented herein. However, she also presents a technique for automatically building LCS-based lexicons for Arabic, Korean, and Spanish, using bilingual dictionaries, LDOCE,

10 The Swedish SIMPLE Lexicon, `http://spraakdata.gu.se/simple/swedish.simple.lexicon.html`. Accessed May 22, 2009.

11 The *Gothenburg Lexical Data Base*, `http://spraakbanken.gu.se/gldb/`. Accessed May 22, 2009.

and WordNet, which at least should be mentioned. Her method uses semantic information about synonymy between verbs, from WordNet, combined with syntactic information from LDOCE, to classify English verbs from the glosses in the bilingual dictionaries according to Levin's (1993) verb classes. Based on the principle that synonymous verbs share distributional patterns[12], Dorr's method constructs LCS lexicon entries for new English verbs found in the glosses based on existing semantic definitions for already classified, synonymous verbs. The non-English verbs were then mapped to LCS entries on the basis of the English glosses in the bilingual dictionaries (Dorr et al. 1995; Dorr 1997).

2.4.4 *Triangulation*

Kay (1997) and Chen et al. (2008) present triangulation, or transitive translation, a method for improving machine translations. The idea behind triangulation can be explained through an example. Consider the task of translating a sentence, for example, from Spanish to English. Assume that the Spanish sentence, S, due to different kinds of ambiguities is translated into the English candidate sentences E_1, E_2, and E_3. Furthermore, assume that S also can be translated into a third language, for example German, as G_1 and G_2. Then, if either G_1 or G_2 can be translated to E_3, intuitively there is a good chance that E_3 is a correct translation of S.

The principle behind triangulation does not necessarily have to involve a third language. Instead of a focusing on languages, the same principle can be applied to different analysis methods for translation from one language to another. For example, given two completely different ways of analyzing an input sentence that each produce one set of candidate translated sentences, if these sets are overlapping, then the sentences that occur in both sets are likely to be correct. As argued by Chen et al. (2008), if we assume that the non-overlapping translated sentences are caused by "misleading information, or noise" in the translation process, "there is no reason to expect the noise in the two systems to correlate strongly."

Even though Kay and Chen et al. apply triangulation to sentence-level translations, the idea can be transferred to word-level translations too. One can argue that the method I present in Chapter 4 indirectly makes use of the driving forces behind the triangulation principle.

12 The distributional pattern of a verb, the range of grammatical environments it can occur in, defines the verb's argument structures and thereby also its semantics.

2.4.5 *Norwegian Semantic Knowledge Bases*

In the fall of 2002, when I began this research, there were no existing semantic resources like WordNet for Norwegian.

Subsequently, a couple of projects have attempted to automatically generate lexical semantic resources by different means. Dyvik (2004) has developed a method that automatically builds a semantic resource based on the implicit semantic information present in parallel corpora. Later, Nygaard (2006) developed a Norwegian wordnet by parsing and analyzing the definitions of words in the (monolingual) Norwegian machine-readable dictionary *Bokmålsordboka* (Wangensteen 2005). Both of these approaches were discussed in Section 1.1, and I will discuss the results of my work in relation to these projects in sections 4.11.4 and 6.4.

Open-Domain Question Answering

Another Norwegian semantic resource, which is currently being manually developed, is a Norwegian part of the SIMPLE project (Lenci et al. 2000). Although the Norwegian SIMPLE resource will provide much interesting semantic information, its coverage will be rather restricted, due to its limited size of approximately 10,000 lemmas.

2.5 Open-Domain Question Answering

The field of QA is concerned with enabling computers to automatically answer questions expressed in natural language. The questions can be simple or complex, and the answer should be precise according to the knowledge available to the system. If a complete answer is not readily available, the system should be able to deduce an appropriate answer from relevant available knowledge (Green and Raphael 1968).

This is not the same as IR systems—like Web search engines—do, because IR systems in general only provide the user with a list of documents that the system deems relevant to the user's query, based on a combination of pattern matching and statistical measures. Admittedly, some search engines incorporate, for example, lexical semantic knowledge in their algorithms, but in general they do not apply any deep analyses, like text interpretation or knowledge acquisition, to the text.

In contrast to IR systems, that often provide snippets that highlight relevant passages of the relevant documents in their search results, the main response from a QA system should ideally be a concise, relevant, and correct answer. However, for improved usability—and increased credibility—a QA system might also provide references to the sources of the knowledge used to generate such an answer.

As one can see from the above requirements, a QA system has to apply sophisticated NLP techniques to analyze the user's question, to understand what the user wants, and further to "deduce that a certain string in a document sentence, or composition of strings from multiple sentences, is the correct answer to the question" (Grunfeld and Kwok 2006).

Research on natural language QA systems has a long history, dating back to 1959 (Simmons 1965). Most of the previous attempts on QA were designed to support only restricted domains, such as SHRDLU by Winograd (1971), a system that was able to manipulate, and reason about the state of, objects in its own "Block world" according to commands and questions given in English; Chat-80 by Warren and Pereira (1982), a knowledge-based system that was able to answer relatively complex questions posed in English about world geography; a system by Gambäck and Ljung (1993) based on the *Swedish Core Language Engine* (S-CLE) that could answer questions about Swedish sovereigns; and BusTUC, to be presented in Section 2.7.1, by Amble (2000).

The last decade there has been growing interest in what is known as open-domain question answering (open-domain QA), both academically and commercially (Paşca and Harabagiu 2001; Paşca 2003; Strzalkowski and Harabagiu 2006). In contrast to traditional QA systems, designed for particular domains, open-domain QA systems are not restricted to any particular domain.

However, lifting the single-domain restriction has several important consequences. For example, the vocabulary actively used both in source documents and by the system's users is probably increased, demanding larger lexicons. The same situation might hold for the grammar, but as pointed out by Moldovan and Rus (2001); Moldovan et al. (2003), their top ten most frequently used grammar rules cover 90 % of their test cases for WordNet glosses.[13] If the system should be able to interpret the text of the source documents, it will probably also need a larger world model.

Furthermore, in open-domain QA the volumes of text are usually much larger than in traditional QA. Therefore, since around the start of the new millennium there has been a trend (Paşca and Harabagiu 2001; Moldovan et al. 2006) to favor hybrid solutions that combine techniques from several fields of NLP to build scalable and robust open-domain QA systems over systems purely based on text interpretation. For example, in such hybrid solution systems, being able to perform efficient and reliable named-entity recognition becomes crucial as the

13 Moldovan and Rus therefore call this the 10–90 rule.

volume of text continues to grow. Likewise, methods from IR can be used to efficiently find relevant passages of text that might contain (parts of) the answer to a question. Furthermore, techniques from IE can be of help during extraction of answers from candidate passages.

Open-Domain Question Answering

However, to adequately answer questions, the QA systems still need different kinds of world knowledge and the ability to reason with it. As said by Lenat (2006), demonstrating QA through logical deduction from rules and facts in the CYC knowledge base:

> "So, if you have these pieces of knowledge, finding this match is trivial. If you don't have these pieces of knowledge, finding this match is impossible. It's not like you can add another 15,000 servers, or let your algorithm run another five seconds, and it would find this match. It will never find this match without these pieces of knowledge."

2.5.1 *Deep Analyses versus Wide Coverage*

NLP technologies usually force a trade-off between wide coverage and depth of analysis. For example, many statistical and machine-learning approaches to NLP are based on detecting and counting occurrences of words and phrases, while other NLP approaches are based on the use of grammars and knowledge about the world to analyze the complete structure of a phrase, sentence, or passage. The word-occurrence approaches usually have no idea of each word's or phrase's semantics, but are able to find relevant information by detecting frequent patterns or using statistically motivated matching rules. Such approaches often offer a wide coverage at the cost of a shallow analysis leading to higher levels of recall, while approaches that perform deep analyses typically offer higher levels of precision at the cost of a narrow coverage. This trade-off between wide-but-shallow and deep-but-narrow affects all NLP systems.

Based on the tradeoff between wide coverage and depth of analysis, it should be no surprise that NLU systems, able to interpret and reason about textual content through deep analyses, generally are limited by a narrow coverage. Despite this, one of the goals for research on open-domain QA is to be able to create systems that precisely answer questions, stated in natural language, that are not restricted to any particular domain. This struggle to achieve both deep analyses *and* wide coverage makes open-domain QA a particular interesting application of ontologies.

2.6 Encyclopedic Question-Answering Systems

A particular application of open-domain QA is within the domain of acquiring knowledge from, and answering questions about, the contents of encyclopedic texts. TUClopedia, described in Chapter 5, is not the first attempt to create such a system.

For example, earlier research on automatic acquisition of knowledge from encyclopedic texts includes the MURAX system (Kupiec 1993), which used only a *shallow* natural language analysis in order to answer Trivial Pursuit questions. The SNOWY system, on the other hand, was able to acquire knowledge from unedited texts from the *World Book Encyclopedia* (1987) within the rather narrow domain of "the dietary habits of animals, their classifications and habitats" and to answer an ample set of questions about the contents of the knowledge database (Gomez 1994). Another attempt, reported by Hull and Gomez (1999), targets a somewhat wider domain through automatically acquiring *biographic* knowledge from articles in the *World Book Encyclopedia*. However, the domain of biographic knowledge is still not as wide, or open, as the domain defined through general encyclopedia articles.

Nonetheless, all of the systems just mentioned handle questions posed in English only. On the other hand, the systems presented by Gambäck and Ljung (1993) and Jönsson et al. (2004) that are able to answer questions about Swedish sovereigns and Ornithology, respectively, could perhaps be considered to constitute *encyclopedic* QA systems for a Scandinavian language, but only for very narrow domains.

2.7 Natural Language Processing Systems for Norwegian

There are three major natural language processing systems for Norwegian. These are *The Understanding Computer* (TUC), *Norwegian Resource Grammar* (NorSource) and *Norsk komputasjonell grammatikk* (NorGram). Below, a short presentation of each will be given with respect to the semantic resources they use.

2.7.1 *The Understanding Computer (TUC)*

TUC (Amble 2000; Amble et al. 2002) is a NLU system designed to be easily adaptable to new applications and domains. The system is bilingual—it understands both Norwegian and English—but the core of TUC uses a language-independent logic to represent knowledge. For each of the two languages, TUC contains a lexicon, morphology rules

and a grammar[14]. In addition, TUC contains a semantic knowledge base, and modules for interfacing external sources of information, e.g. Structured Query Language (SQL) databases.

Earlier TUC-based systems, still subject both to research and development, have successfully been applied to several narrow domains. The systems include BusTUC (Amble 2000), a natural-language based expert system route advisor for the public bus transport in Trondheim, Norway (in commercial use[15]), and GeneTUC (Sætre 2006), an application of TUC to the domain of understanding and extracting information about gene and protein interactions from medical articles. Furthermore, Bruland (2002) developed a system called ExamTUC that could extract information from an article about the kings of Norway in order to automatically grade written answers to examination questions based on the same article. This was an early attempt at adapting TUC to a broader domain.

TUC uses a semantic network both to filter out semantically invalid candidate interpretations when it parses text and for reasoning about knowledge in order to answer questions posed by users. Hence, TUC will fail to interpret any sentence that contains concepts not covered by TUC's semantic network; that is, concepts not part of the semantic network's domain. Besides, when TUC answers a question, the quality of the answer will depend on both the coverage and the quality of the semantic network.

Not surprisingly, the scopes of the semantic networks previously developed for TUC reflect the narrow domains TUC has been applied to.

TUC's Grammar

The grammar in TUC uses a Context-Sensitive-Categorial-Attribute-Logic (ConSensiCAL) grammar formalism. This formalism is an easy-to-use variant of a generalization of Definite Clause Grammar (DCG) called Extraposition Grammar (Pereira and Warren 1980).

Where a Context-Free Grammar (CFG) uses a rule skeleton like $A \rightarrow \gamma$, stating that a nonterminal A can expand into an arbitrary string of terminal and nonterminal symbols, a Context-Sensitive Grammar (CSG) uses a rule skeleton like $\alpha A \beta \rightarrow \alpha \gamma \beta$, where α and β represent arbitrary strings of terminal and nonterminal symbols, just like γ. Thus,

14 Based on experiences with BusTUC, Amble (2000) states that the grammars are surprisingly similar, but no effort has been made to coalesce them.

15 The commercial version of BusTUC is on-line at `http://www.team-trafikk.no/`, while a noncommercial version is available at `http://www.idi.ntnu.no/~tagore/bustuc/`.

the context-sensitive features of the ConSensiCAL formalism mean that the parser can use information about the context of the nonterminal in the left-hand side of the rules[16].

The categorial part of the formalism enables new syntactic categories to be derived from the basic categories by the application of one or more of three operations.

Finally, the attribute-logic part of ConSensiCAL means that the categories of the grammar can be augmented by attributes that are subject to logical restrictions.

For further explanation of the ConSensiCAL grammar formalism, see (Amble 2003). The grammar is basically a grammar for simple statements, while questions are derived by the use of movements.

TUC's Parser

During parsing, TUC performs semantic type checking. Therefore, TUC is able to discard candidate parses as soon as they are recognized as semantically invalid by this check. Hence, TUC's parser will only accept sentences that are syntactically, grammatically, and semantically valid.

Amble (2000; p. 4) states that during disambiguation between candidate interpretations TUC applies a heuristics based on the idea that "the longest possible phrase of a category that is semantically correct is in most cases the preferred interpretation"; a guideline he notices has proved "almost irreproachable."

When TUC has successfully parsed a sentence (simple statement or question), the sentence is transformed into a TUC Query Language (TQL) expression, which represents the information in a first-order event calculus (Kowalski and Sergot 1986). At the core of the construction of the TQL expressions is the use of verbal complements where the event functions as a link between the semantic units that comprise the newly acquired knowledge.

2.7.2 *The LOGON Project*

The *Leksikon, ordsemantikk, grammatikk og oversettelse for norsk* (LOGON) project "aims to deliver high-quality, document-level Norwegian–English MT based on the combination of a symbolic, semantic-transfer-oriented backbone and stochastic processes for ambiguity management and robustness" (Oepen et al. 2004).

The syntactic analysis of the Norwegian input is handled by the Norwegian Lexical-Functional Grammar (LFG) resource grammar Nor-

16 Actually, the rule skeleton for the ConSensiCAL grammar formalism is $\alpha A \beta \rightarrow \gamma$.

Gram developed by Dyvik (1999). The translation is done by transferring a Minimal-Recursion Semantics (MRS) representation (Copestake et al. 2005) of the Norwegian analysis into an English MRS representation, based on a set of MRS transfer rules. Next, the Head-Driven Phrase Structure Grammar (HPSG) based English Resource Grammar (Flickinger 2000) from the *Linguistic Grammars Online* (LinGO) project is used for generation of natural language output in English.

Natural Language Processing Systems for Norwegian

Another resource grammar for Norwegian is the HPSG-based NorSource (Hellan and Haugereid 2003). The grammar currently handles most Norwegian *grammatical constructions*, but contains enough semantic information to generate interesting semantic parses only for approximately 1,000 words[17].

Both NorGram and NorSource are based on grammar formalisms that are based on unification of feature structures (FSs). These FSs can contain semantic information, such as which nominals are arguments to which verbs, but these semantic features do not refer to a world model, or ontology.

17 Lars Hellan, phone conversation with Martin Thorsen Ranang, 21st of July, 2006.

—Compounding is probably the most important and widely used word formation process in Norwegian.

Johannessen (2001)

3 Handling of Norwegian Compounds

This chapter presents a practical implementation, including a complementary specification, of an algorithm for automatic analysis and treatment of Norwegian compound words.

3.1 Motivation

Because Verto—the implementation of the method for automatically mapping content words in a non-English source language to WordNet senses, to be presented in the next chapter—depends heavily on automatically looking up words in a bilingual dictionary, handling compounds not already covered by the dictionary could greatly increase the coverage of the method. Furthermore, because of the importance of compounding as a word formation process, both in Norwegian and at least some of the other Germanic languages—such as German, Dutch, Danish, Swedish, and Icelandic—the method should be able to handle mapping of compounds.

However, since automatically analyzing compounds is a nontrivial task in itself, this chapter focuses on that topic in isolation, before we delve into the main method itself in Chapter 4. That way, this chapter constitutes important building blocks for the method to be presented in the *next* chapter.

3.2 Norsk Komputasjonelt Leksikon (NorKompLeks)

The only important externally developed resource in the compound-word analysis module is a Norwegian lexicon called *Norsk komputasjonelt leksikon* (NorKompLeks) (Nordgård 1998). NorKompLeks contains information about lexical class, inflection, stems (or base forms), phonology, and argument structures for Norwegian words.

Guiding Principles

The compound-word analyzer interfaces NorKompLeks through a library module. For each keyword looked up through the NorKompLeks module, the returned information is a list of tuples. The list contains one element per inflected form that the keyword may represent. Each tuple consists of a stem, a set of attributes, and a number identifying the lexeme. For example, the returned information for the word *«ansikt»* ("face") is

$$\begin{aligned}\textsc{NorKompLeks}(\text{ansikt}) = \\ \langle(\text{ansikt}, \{(\text{form}, \text{ind}), (\text{gend}, \text{n}), \\ (\text{num}, \text{pl}), (\text{pos}, \text{subst})\}, 2067) \\ (\text{ansikt}, \{(\text{form}, \text{ind}), (\text{gend}, \text{n}), \\ (\text{num}, \text{sg}), (\text{pos}, \text{subst})\}, 2067)\rangle.\end{aligned} \tag{3.1}$$

As can be seen from (3.1), the word form *«ansikt»* can be both the singular and plural indefinite form of the neuter noun (*«substantiv»* in Norwegian) with the stem *«ansikt»*. However, the form *«ansiktet»* ("face.the") yields

$$\begin{aligned}\textsc{NorKompLeks}(\text{ansiktet}) = \\ \langle(\text{ansikt}, \{(\text{form}, \text{def}), (\text{gend}, \text{n}), \\ (\text{num}, \text{sg}), (\text{pos}, \text{subst})\}, 2067)\rangle,\end{aligned} \tag{3.2}$$

which shows that it is the singular definite form of the same lexeme.

The only information required by the presented method is the lexical class and stem of keywords. The other information is ignored.

3.3 Guiding Principles

In the implementation of the automatic compound-word analyzer described herein, the following guidelines for automatically analyzing compound words, quoted from Johannessen and Hauglin (1996), are incorporated into the *algorithms*:[1]

> (JH17) Lexical compounding is preferable to compounding with epenthetic phones.
>
> [with the exceptions]

1 I have changed the labels. Each of the quoted principles are identified by a label (JH*n*), where JH refers to the authors, Johannessen and Hauglin, while *n* refers to the label used in (Johannessen and Hauglin 1996).

(JH20) Epenthetic -s- is preferred to lexical compounding when the -s- can be ambiguous between epenthetic use and the first letter of a verbal last member.

(JH25) Epenthetic -s- is preferred to lexical compounding when the first member is itself a compound.

(JH30) If two analyses have the same number of members and there is no epenthesis involved, choose the one, if any, that is a noun.

(JH33) If two analyses are equal with respect to epenthesis and part of speech, and one has a first member that is itself a compound, then choose that one.

(JH48) Choose the analysis (or analyses) with the fewest compound members.

While the following principles—also presented in the same paper—are enforced through the design of the parser's *grammar*:

(JH22) Epenthetic -s- can only follow noun stems.

(JH27) Epenthetic -s- cannot follow epenthetic -e- and vice versa.

(JH35) Epenthetic -e- can only be attached to a stem that is monosyllabic.

(JH37) Other possible stems can be prior to the stem preceding the -e-, if they do not form a compound with that stem.

(JH40) Epenthetic -s- cannot occur after a sibilant or a final consonant sequence containing a sibilant.

(JH42) Except when the consonant belongs to a compound.

However, the guideline

(JH46) If the first member is unknown, choose the analysis with the longest last member.

is not used, because the analyzer implemented herein requires that each of the constituents is either defined through the grammar, like the epenthetic -e- and -s-, or found in the NorKompLeks lexicon.

As shown above, selecting the correct analysis of how a word is compounded is a nontrivial task. In Section 6.1 I will discuss how the resulting resource generated by the method described herein may be used to improve the compound-word analysis.

Algorithm 3.1: Function that returns all valid compound-word analyses sorted by preference.

```
1: function Get-Compounds-Sorted(word)
2:     analyses ← Get-Compounds-Unsorted(word)
3:     prioritized_analyses ← Prioritize-Analyses(analyses)
4:     sorted_analyses ← Sort(prioritized_analyses)
5:     return sorted_analyses
```


3.4 Algorithms for Automatic Analysis of Compounds

This section will present the algorithms developed for doing the automatic compound-word analysis. In the following algorithm descriptions, the functions Map[2] and Filter[3] known from functional programming are assumed available.

The main function of the automatic compound-word-analyzer module is Get-Compounds-Sorted, shown in Algorithm 3.1. The algorithm gives an overview of how the analyzer works. If NorKompLeks cannot directly look up the keyword, behind the scenes the module will call Get-Compounds-Sorted with the string that represents the keyword to be analyzed. The returned value is a list containing theoretically valid analyses, ordered by their likelihood of being relevant and correct.

2 The Map(f, seq) function takes a function, f, and a sequence, $seq = \langle x_1, x_2, \ldots \rangle$, as arguments. The return value is a new sequence that contains the values of $f(x)$, where $x \in seq$.

3 The Filter(f, seq) function takes a function, f, and a sequence, $seq = \langle x_1, x_2, \ldots \rangle$, as arguments. The return value is a new sequence that only contains the elements of seq for which $f(x)$, where $x \in seq$, evaluates to *True*.

For example, given the Norwegian compound «*musikkspiller*» ("music player")—which is not an entry in NorKompLeks—the output from the function call begins with

Handling of Norwegian Compounds

$$
\begin{aligned}
&\textsc{NorKompLeks.Get-Compounds-Sorted}(\text{musikkspiller}) = \\
&\quad\langle\langle(\text{musikk},\\
&\qquad\langle(\{(\text{pos},\text{subst}),(\text{num},\text{sg}),\\
&\qquad\quad(\text{form},\text{ind}),(\text{gend},\text{m})\},\text{musikk},39595)\rangle),\\
&\quad(\text{spiller},\\
&\qquad\langle(\{(\text{pos},\text{subst}),(\text{num},\text{sg}),\\
&\qquad\quad(\text{form},\text{ind}),(\text{gend},\text{m})\},\text{spiller},56073)\rangle)\rangle,
\end{aligned}
\tag{3.3}
$$

which correctly represents the "music player" interpretation, followed by

$$
\begin{aligned}
&\langle(\text{musikk},\\
&\quad\langle(\{(\text{pos},\text{subst}),(\text{num},\text{sg}),\\
&\qquad(\text{form},\text{ind}),(\text{gend},\text{m})\},\text{musikk},39595)\rangle),\\
&\quad(\text{s},\langle(\{(\text{pos},\text{joint})\},\text{epenthetic-s},-1)\rangle),\\
&\quad(\text{piller},\\
&\quad\langle(\{(\text{pos},\text{subst}),(\text{num},\text{pl}),\\
&\qquad(\text{form},\text{ind}),(\text{gend},\text{m})\},\text{pille},44933),\\
&\qquad(\{(\text{pos},\text{subst}),(\text{num},\text{pl}),\\
&\qquad(\text{form},\text{ind}),(\text{gend},\text{f})\},\text{pille},44934),\\
&\qquad(\{(\text{pos},\text{subst}),(\text{num},\text{pl}),\\
&\qquad(\text{form},\text{ind}),(\text{gend},\text{m})\},\text{pille},44934)\rangle)\rangle,
\end{aligned}
\tag{3.4}
$$

Algorithm 3.2: Function that returns all valid compound-word analyses.

```
1: function Get-Compounds-Unsorted(word)
2:     analyses ← ⟨⟩
3:     for parts ∈ Split-Word(word) do
4:         trees ← Get-Complete-Word-Parses(parts)
5:         analyses ← analyses +Map(Analyze-Parse, trees)
6:     return analyses
```


which translates to "music pills", and

$$\begin{aligned}&\langle(\text{musikk},\\&\quad\langle(\{\langle(\text{pos},\text{subst}),(\text{num},\text{sg}),\\&\qquad(\text{form},\text{ind}),(\text{gend},\text{m})\rangle\},\text{musikk},39595)\rangle),\\&\quad(\text{spiller},\\&\quad\langle(\{\langle(\text{vform},\text{pres}),(\text{pos},\text{verb})\rangle\},\text{spille},56059)\rangle)\rangle,\\&\quad\ldots,\rangle\end{aligned} \tag{3.5}$$

which represents the verbal interpretation "music plays" followed by less and less probable analyses.

As can be seen from these examples, the format of the output from Get-Compounds-Sorted is slightly different from the output of the NorKompLeks module shown in (3.1) and (3.2). The difference is mainly that the NorKompLeks "stem" is included between the set of attributes and the NorKompLeks id for the lexeme. The reason for this is to make it easier to evaluate the suggested compound-word analyses and that the output format seen in (3.1) and (3.2) can easily be generated for any of the single cases (3.3), (3.4), or (3.5).

The two most important components of the Get-Compounds-Sorted method are Get-Compounds-Unsorted and Prioritize-Analyses. Below, each of these functions, and the functions they rely on, will be described.

3.4.1 *The* Get-Compounds-Unsorted *Algorithm*

The Get-Compounds-Unsorted method, shown in Algorithm 3.2, is responsible for generating a list, named *analyses*, of the valid compound-word analyses of *word*.

Algorithm 3.3: Function for suggesting chunking, or segmentation, of compound words.

```
1:  function Split-Word(word, interpretations ← ∅)
2:      if word ∈ interpretations then
3:          return interpretations[word]
4:      interpretations[word] ← ⟨⟩                          ▷ Empty list.
5:      if |Get-Part-of-Word-Parses((word))| > 0 then
6:          interpretations[word]+ ← ⟨(word)⟩
7:      for all i ∈ [(|word| − 1), 1] do
8:                      ▷ Generate combinations of word split at position i:
9:          for all (first, last) ∈ Combine-Parts(word[: i], word[i :]) do
10:             if |Get-Part-of-Word-Parses((last))| = 0 then
11:                 continue                          ▷ . . . the surrounding loop.
12:             for all suggested_start ∈ Split-Word(first, interpretations)
    do
13:                 interpretations[word]+ ← ⟨suggested_start + (last)⟩
14:     return interpretations[word]
```

Algorithm 3.4: Function that returns all valid parse trees given *parts* as input, filtering out sequences that cannot successfully be parsed as complete words.

```
1:  function Get-Complete-Word-Parses(parts)
2:      if Is-Assumed-Cruft(parts) then
3:          return ⟨⟩
4:      trees = Parse(parts)
5:      return Filter(Is-Complete-Word, trees)
```

ALGORITHM 3.5: Function that returns all valid parse trees given *parts* as input, filtering out sequences that cannot successfully be parsed as a part-of-word.

```
1: function GET-PART-OF-WORD-PARSES(parts)
2:     if IS-ASSUMED-CRUFT(parts) then
3:         return ⟨⟩
4:     trees = PARSE(parts)
5:     return FILTER(IS-PART-OF-WORD, trees)
```

First, SPLIT-WORD, shown in Algorithm 3.3 on the preceding page, returns a list of possible ways to segment *word*. Second, for each of the segmentations, GET-COMPLETE-WORD-PARSES, shown in Algorithm 3.4, returns a—possibly empty—list of parse trees, each representing a valid analysis of the compound represented by the segmentation. Third, each tree is separately analyzed by calling the ANALYZE-PARSE function. The function takes a single parse tree as an argument and returns a *TreeAnalysis* object that—in addition to keeping a reference to the original parse tree—features a set of methods that can be used to describe different higher-level characteristics of the parse. The analyses are added to the *analyses* list. Finally, the list *analyses* is returned. In the following the workings of these algorithms will be explained.

SPLIT-WORD is a recursive function that repeatedly calls itself with different parts of *word* and checks whether they are possible to parse as parts of a word by calling GET-PART-OF-WORD-PARSES, shown in Algorithm 3.5. A central piece of SPLIT-WORD is the for-construct on lines 7–9 that, given the input string *word* and—for now—ignoring the call to COMBINE-PARTS, generates the sequence

$$\langle (word[:i], word[i:]) | i \in [(|word| - 1), 1] \rangle. \tag{3.6}$$

Given that *word* equals `"musikkatalog"`, this sequence expands to

$$\begin{aligned} \langle &(\texttt{"musikkatalo"}, \texttt{"g"}), (\texttt{"musikkatal"}, \texttt{"og"}), \\ &(\texttt{"musikkata"}, \texttt{"log"}), (\texttt{"musikkat"}, \texttt{"alog"}), \\ &(\texttt{"musikka"}, \texttt{"talog"}), (\texttt{"musikk"}, \texttt{"atalog"}), \\ &(\texttt{"musik"}, \texttt{"katalog"}), (\texttt{"musi"}, \texttt{"kkatalog"}), \\ &(\texttt{"mus"}, \texttt{"ikkatalog"}), (\texttt{"mu"}, \texttt{"sikkatalog"}), \\ &(\texttt{"m"}, \texttt{"usikkatalog"}) \rangle. \end{aligned} \tag{3.7}$$

ALGORITHM 3.6: Function that returns a Boolean value defining whether the sequence of words given as the argument is "cruft" or not.

```
1: function IS-ASSUMED-CRUFT(parts)
2:     return (|parts| > 2) ∧ ( ( Σ_{part∈parts} |part| ) / |parts| < 3 )
```

ALGORITHM 3.7: Function for combining parts of a word split in two.

```
1: function COMBINE-PARTS(a, b)
2:     if (|a| ≥ 2) ∧ (a[−1] ∈ consonants) ∧ (a[−2] = a[−1]) then
3:         return ⟨(a, b), (a, a[−1] + b)⟩
4:     else
5:         return ⟨(a, b)⟩                    ▷ The obvious combination.
```

To avoid duplicate processing, the SPLIT-WORD algorithm implements memoization through the variable *interpretations* that cache earlier results. The first time the function is called, with the input *word* as the only argument, *interpretations* assumes ∅ as a default value. However, in the consecutive recursive calls *interpretations* is supplied as a second argument. It should be noted that the use of COMBINE-PARTS, shown in Algorithm 3.7, ensures that in some cases additional letters can be added to a segment. How this is done is described below.

GET-PART-OF-WORD-PARSES is a function that basically forwards its only argument, *parts*, to the parser described below, in Section 3.5. The returned parse trees are filtered through IS-PART-OF-WORD, a function that checks the root node to see whether the tree represents something that can be considered part of a word (the alternative would be a complete word). However, the function also calls the function IS-ASSUMED-CRUFT, shown in Algorithm 3.6, to filter out word segmentations that are considered noise, or "cruft". A segmentation is considered noise if it consists of more than two parts and the average length of the parts is shorter than a given constant[4].

4 Based on experiences during the development of these algorithms, the constant is defined to be 3.

A Norwegian spelling rule states that there should never be more than two contiguous occurrences of the same consonant in any word. For example, the naïve compounding of «*musikk*» ("music") and «*katalog*» ("catalog") through simple concatenation,

(*) «*musikkkatalog*» ("music catalog"), (3.8)

results in an error. The correct way to compound these words is to simply ignore the third contiguous consonant occurrence, as in

«*musikkatalog*» ("music catalog"). (3.9)

This "consonant shortening over compound boundaries" was identified as a potential problem by Karlsson (1992) too. However, to avoid overgeneration of analyses in SWETWOL, he chose to simply list the roughly 360 compounds of that type he had identified.

The COMBINE-PARTS function always returns a sequence containing its two arguments, a and b, paired together as a tuple, (a, b). However, to counter the contiguous-consonants rule mentioned above, the function also checks whether the two last characters of a represent the *same consonants*. In that case, the function includes a tuple $(a, a[-1] + b)$, where the last character of a has been added to the beginning of b, in the result list. For example, given that the input word was «*musikkatalog*», at some point during the evaluation of SPLIT-WORD("musikkatalog"), the call

$$\text{COMBINE-PARTS}(\texttt{"musikk"}, \texttt{"atalog"}) = \langle(\texttt{"musikk"}, \texttt{"atalog"}), (\texttt{"musikk"}, \texttt{"katalog"})\rangle \tag{3.10}$$

will be made. In this case, the first tuple will not parse as neither a part of a word, nor as a complete word. Thus, the introduction of the second tuple actually plays a key role in finding valid analyses of the compound. For example, the expression in (3.7) that shows the different first-level segmentations that are considered during the analysis of «*musikkatalog*» does not contain ("musikk", "katalog"). However, since SPLIT-WORD utilizes COMBINE-PARTS, the set of considered segmentations becomes

$$\begin{aligned}\langle&(\texttt{"musikkatalo"}, \texttt{"g"}), (\texttt{"musikkatal"}, \texttt{"og"}),\\ &(\texttt{"musikkata"}, \texttt{"log"}), (\texttt{"musikkat"}, \texttt{"alog"}),\\ &(\texttt{"musikka"}, \texttt{"talog"}), (\texttt{"musikk"}, \texttt{"atalog"}),\\ &(\texttt{"musikk"}, \texttt{"katalog"}), (\texttt{"musik"}, \texttt{"katalog"}),\\ &(\texttt{"musi"}, \texttt{"kkatalog"}), (\texttt{"mus"}, \texttt{"ikkatalog"}),\\ &(\texttt{"mu"}, \texttt{"sikkatalog"}), (\texttt{"m"}, \texttt{"usikkatalog"})\rangle.\end{aligned} \tag{3.11}$$

Algorithm 3.8: Function that returns all valid compound-word analyses rated by order of preference.

```
1: function Prioritize-Analyses(analyses)
2:     prioritizations ← ⟨⟩
3:     for analysis ∈ analyses do
4:         prioritizations ← prioritizations + ⟨Prioritize(analysis, analyses)⟩
5:     return prioritizations

6: function Prioritize(analysis, analyses)
7:     return (analysis.Get-Number-of-Compounds(),
                − analysis.Has-Preferable-Epenthetic-S(),
                − analysis.Has-Adjective-Noun-Ambiguity(analyses),
                analysis.Get-POS-Weight-Mapping()[−1],
                analysis.Has-Suffix-Head(),
                − analysis.First-Member-Is-Compound(),
                − analysis.Has-Ambiguous-Epenthetic-E(analyses),
                analysis.Contains-Inflection-Formatives(),
                analysis.Contains-Epentheses(),
                − analysis.First-Member-Is-Probably-Compound(analyses),
                analysis.Get-Reversed-Negated-Part-Lengths(),
                analysis.First-Member-Is-Verbal(),
                analysis.Get-POS-Weight-Mapping-Adjusted(),
                analysis)
```

Handling of Norwegian Compounds

The first step of the for-loop in the Get-Compounds-Unsorted function is the Get-Complete-Word-Parses function, shown in Algorithm 3.4. The function filters out noisy input and passes on its input argument to the parser, just like Get-Part-of-Word-Parses. However, the returned list contains only parse trees of *complete (compound) words*, according to the parser's grammar.

The second step of the for-loop is to create a new list by applying the function Analyze-Parse to each parse tree in turn. The Analyze-Parse function basically wraps each parse tree in a *TreeAnalysis* object by calling the class' constructor and returns the newly created object.

3.4.2 *The* Prioritize-Analyses *Algorithm*

After Get-Compounds-Unsorted has returned a list of analyzed valid parses, the Get-Compounds-Sorted function (Algorithm 3.1) will first make a prioritized version of the list and then return a sorted version of it. This section will present how the prioritization is done.

The Prioritize-Analyses function, shown in Algorithm 3.8, takes a list of *TreeAnalysis* objects as its input argument. The function returns a list of prioritized analyses built by calling Prioritize—also shown in Algorithm 3.8—for each analysis in the input argument.

When comparing two n-tuples, comparisons are made in pairs; first, the leftmost element from both tuples are compared, next, the second element from both tuples are compared, and so on. The way tuples are compared means that they may easily be used for ordering lists; the lower the value, the higher the priority. The following call to Sort with a list of tuples shows how this works:

$$\begin{aligned}\textsc{Sort}(\langle(3,-2,\texttt{"ab"}),(1,3,\texttt{"b"}),(3,-2,\texttt{"a"}),(-1,8,\texttt{"c"})\rangle) = \\ \langle(-1,8,\texttt{"c"}),(1,3,\texttt{"b"}),(3,-2,\texttt{"a"}),(3,-2,\texttt{"ab"})\rangle.\end{aligned} \tag{3.12}$$

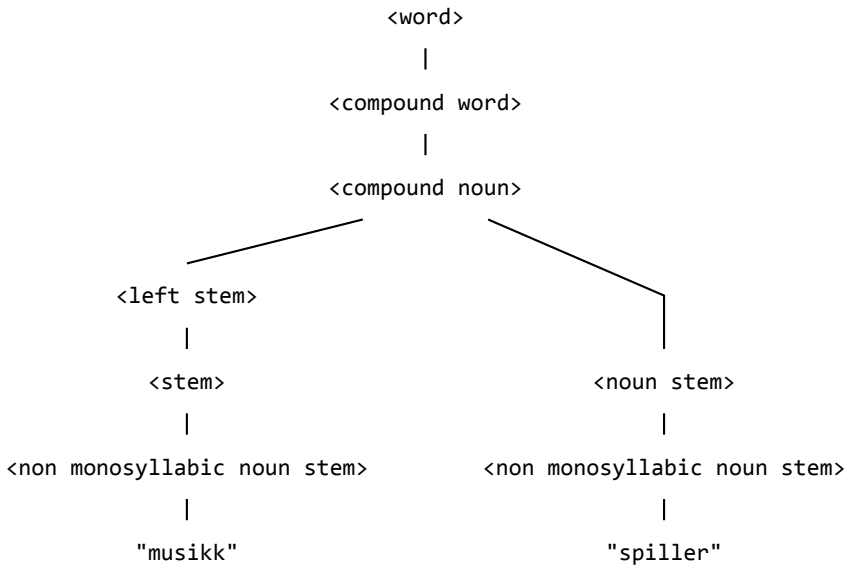
The Prioritize function returns a tuple where all but the last element are numeric elements. The different high-level characterizing methods of the *TreeAnalysis* objects are called to provide the numeric values in the tuple. As can be seen from the function, some of the method's return values are negated. This is done to turn positive values into higher priorities.

Every ith element of the prioritization tuple makes a difference only among the other analyses that have been characterized identically at each position up to $(i-1)$. The nontrivial part is deciding the order of the calls to the analysis-characterizing methods.

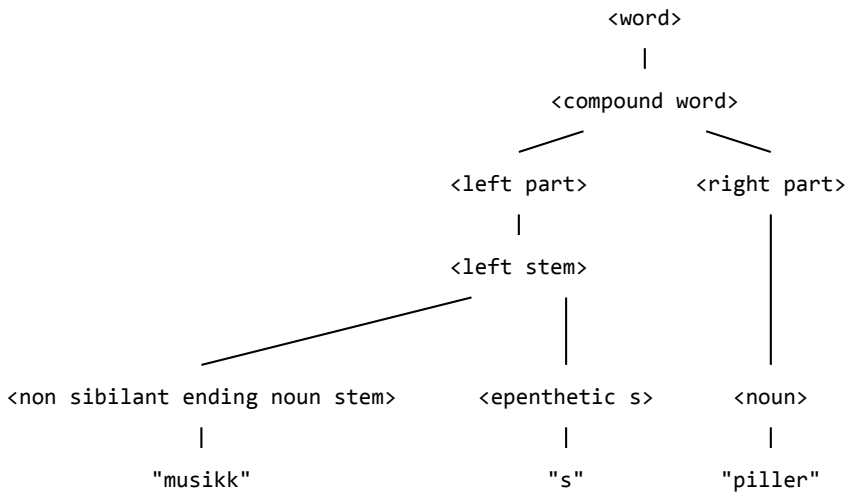
Next, each of the methods used in Prioritize to characterize each analysis will be presented.

The first—most significant—element in the prioritization tuple implements guideline (JH48). The element is simply the number of compounds that comprise the analysis, found by calling *analysis*.Get-Number-of-Compounds(). The function traverses the parse tree and counts compounds. For example, the analyses shown in Figures 3.1a and 3.1b both have a compound count of 1 (the attachment of the epenthetic -s- does not constitute a compound) while the analysis in Figure 3.2 has a compound count of 2.

The second element, supplied by calling Has-Preferable-Epenthetic-S(), is a pragmatic replacement of guidelines (JH20) and (JH25). The



(a) An analysis corresponding to (3.3).



(b) An analysis corresponding to (3.4).

FIGURE 3.1: Two analyses of the compound «*musikkspiller*» returned by the parser and evaluated by the compound-word analyzer.

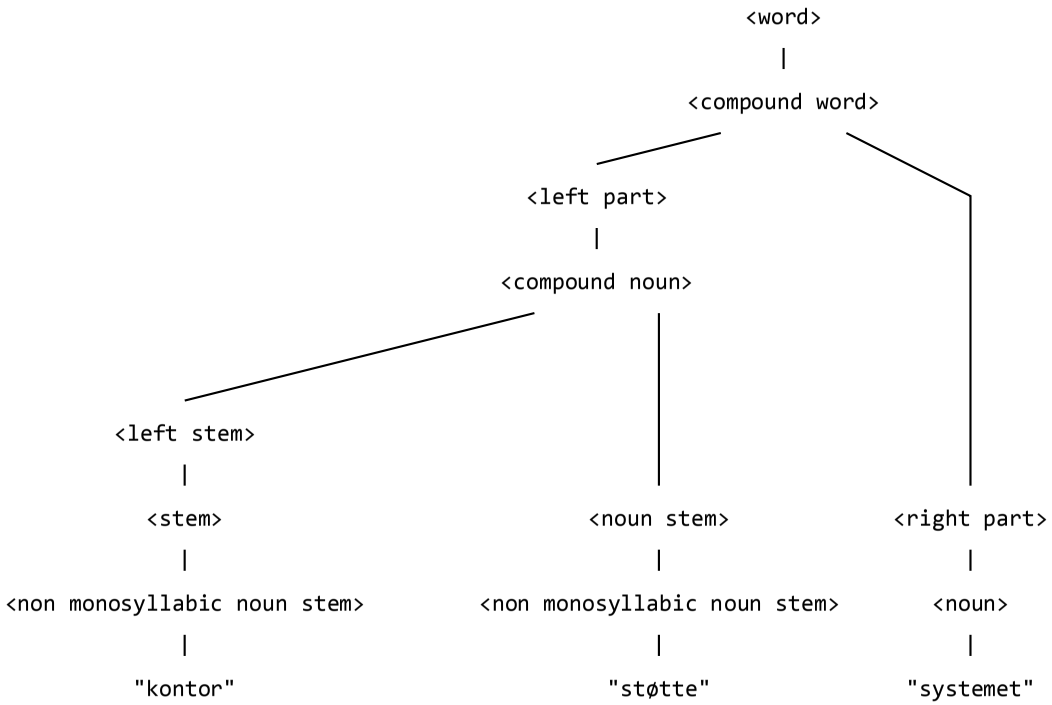


FIGURE 3.2: One of the valid analyses of *«kontorstøttesystemet»* (“office support the.system”).

TABLE 3.1: The weighting of different lexical categories used by the GET-POS-WEIGHT-MAPPING method. A lower value means a higher priority.

Lexical category	Weight	Lexical category	Weight
epenthesis	0	quantifier	3
joint	0	adjective	4
possessive	0	verb	5
noun	1	adverb	6
prefix	2	deflection	10
preposition	3	inflection	10

replacement is motivated by extensive experimentation with the prioritization rules and by testing the analyzer on compound words from real-world texts. The method returns true only if the analysis contains an epenthetic -s- *and* the epenthetic -s- is preceded by a word ending in '-ed' or 'ng'. Since the return value is negated, the following guideline can be put forward:

Guideline 1. *Prefer analyses containing epenthetic -s- only if the epenthetic -s- is preceded by a member ending with '-ed' or '-ng'.*

The third tuple element, obtained by evaluating HAS-ADJECTIVE-NOUN-AMBIGUITY(*analyses*), will give preference to analyses with nominal heads over analyses with adjectival heads. However, this preference is only given if the rest of the analyses are equal with respect to the classification of the terminals. This preference can be seen as a specialization of (JH30) which is also covered by other tuple elements.

The GET-POS-WEIGHT-MAPPING method returns a list of numbers where each number corresponds to a weighting of the lexical category of the different parts of the analysis. The weighting scheme is shown in Table 3.1. For example, the weighting scheme gives higher priority to nouns than to adjectives, and adjectives are given a higher priority than verbs.

The last value in the list returned from GET-POS-WEIGHT-MAPPING is used to define the fourth tuple element; the weighting of the analysis' head, based on its lexical category. Because the weights returned by GET-POS-WEIGHT-MAPPING give higher priority to nouns than to adjectives, the previous tuple element can be seen as an element that counters

that. However, the previous element is only true if the analyses are otherwise equal. Therefore, this element covers (JH30) too.

The fifth value is obtained by calling HAS-SUFFIX-HEAD and is 1 if the head of the analysis is a kind of suffix (compare with the grammar presented in Section 3.5) and 0 otherwise. This value is included to give lower priority to analyses with suffix heads.

The sixth tuple member is -1 if the first member is a compound and 0 otherwise. The value is obtained by calling FIRST-MEMBER-IS-COMPOUND and is used to enforce (JH25). Whether the first member is a compound is determined by checking if the index of the epenthetic -s- in the list of terminals is greater than 1 (starting at 0), which means that there are more than one member to the left of it.

The value of the seventh element is the returned value from the method call HAS-AMBIGUOUS-EPENTHETIC-E(*analyses*). The method implements the preference principle:

Guideline 2. *Epenthetic -e- is preferred when the epenthetic -e- can be ambiguous between epenthetic use after a monosyllabic noun and the last letter of a first-member noun.*

Thus, the tuple element becomes -1 in case the current analysis contains an epenthetic -e- that causes such an ambiguity, and 0 otherwise.

The eighth tuple element, obtained by calling the CONTAINS-INFLECTION-FORMATIVES method, is 1 if the analysis contains any inflection formative at all, and 0 otherwise.

The ninth element, returned by CONTAINS-EPENTHESES(), is 1 if the analysis contains any epenthetic element, and 0 otherwise.

The tenth element, obtained by calling FIRST-MEMBER-IS-PROBABLY-COMPOUND(Analyses), is -1 if the first member is *probably* a compound, and 0 otherwise. This method differs from FIRST-MEMBER-IS-COMPOUND in that it compares the current analysis with the other analyses. Whether the first member is a compound is determined by comparing the first member of the current analysis with concatenations of the first members of the other analyses. This is done in an attempt to detect words that were formed through compounding but have become lexicalized and have thus been included in lexicons and dictionaries. Thus, this tuple element affects the prioritization according to (JH33).

GET-REVERSED-NEGATED-PART-LENGTHS() returns a tuple containing the arithmetically negated length of each of the members of the analysis, in reversed order. Thus, the negated length of the last member of the analysis is the first element of the returned tuple, and so on. The returned tuple constitutes the eleventh element in the prioritization

tuple and is included to give higher priority to analyses with longer members—from right to left.

The twelfth element is returned by a call to First-Member-Is-Verbal() and is simply 1 if the first member is verbal, and 0 otherwise.

The thirteenth element is also a tuple. The tuple is obtained by calling the Get-POS-Weight-Mapping-Adjusted method, which is quite similar to the Get-POS-Weight-Mapping method described above, except that a value of 0.5 is subtracted from the last member if it represents a stem or a gerund.

The final element in the tuple is the analysis object itself, so that it later easily can be extracted from the tuple.

3.5 Compound-Word Parser

The parser implemented in the automatic compound-word analyzer is a modified version of the Earley chart parser (Earley 1970) from the *Natural Language Toolkit* (NLTK) (Bird and Loper 2004; Bird 2006; Bird et al. 2008).

The Earley chart parsing algorithm uses a Context-Free Grammar (CFG) and handles left-recursive grammar rules and ambiguous grammars, while it avoids inefficient reparsing of subtrees through the use of dynamic programming (Jurafsky and Martin 2000; pp. 377–385). Additionally, if more than one parse is valid, the Earley algorithm finds all of them in a single pass through the chart, from left to right. Furthermore, in the general case, given an input of n words, the parser's worst-case behavior is $O(n^3)$, but with an *unambiguous* grammar the worst-case behavior becomes $O(n^2)$ (Earley 1970).

3.5.1 Modifications of the Parser

To be able to adhere to the guidelines proposed by Johannessen and Hauglin (1996) not covered by the algorithms presented in Section 3.4.2, the parser needs knowledge about the terminals' lexical categories, inflection, derivation, and morphological and phonological traits. Much of that information is available in the NorKompLeks, but some information, like the number of syllables in a word, or whether a word ends with a sibilant, is not covered by NorKompLeks.

One way to make all of the information needed for each terminal symbol available to the parser would be to write a script that explicitly generates all of the productions that contain terminal symbols—that

is, the lexicon. For example, the lexicon part of such a grammar would contain rules like

Compound-Word Parser

$$\begin{aligned}
\langle \text{monosyllabic noun stem} \rangle &\rightarrow \texttt{"hopp"} \\
\langle \text{monosyllabic verb stem} \rangle &\rightarrow \texttt{"hopp"} \\
\langle \text{non-monosyllabic noun stem} \rangle &\rightarrow \texttt{"hoppe"} \\
\langle \text{noun} \rangle &\rightarrow \texttt{"hopp"} \\
\langle \text{noun} \rangle &\rightarrow \texttt{"hoppe"} \\
\langle \text{verb} \rangle &\rightarrow \texttt{"hopp"} \\
\langle \text{verb} \rangle &\rightarrow \texttt{"hoppe"},
\end{aligned} \tag{3.13}$$

where each right-hand side of the productions contains static string terminal symbols.

Another way, chosen in this implementation, is to change the parser slightly, so that it accepts grammar productions where the terminal symbols can be general functors, or functional objects. This means, that if a functor, when applied to an input word, returns *True*, then whatever the category the left-hand side of that production represents is warranted by that word. In other words, it becomes possible to perform arbitrary operations on each part of the input during parsing. Thus, the part of the grammar shown in (3.13) instead contains rules like

$$\begin{aligned}
\langle \text{monosyllabic noun stem} \rangle &\rightarrow \textsc{Monosyllabic-Noun-Stem}? \\
\langle \text{monosyllabic verb stem} \rangle &\rightarrow \textsc{Monosyllabic-Verb-Stem}? \\
\langle \text{non-monosyllabic noun stem} \rangle &\rightarrow \textsc{Non-Monosyllabic-Noun-Stem}? \\
\langle \text{noun} \rangle &\rightarrow \textsc{Noun}? \\
\langle \text{verb} \rangle &\rightarrow \textsc{Verb}?,
\end{aligned} \tag{3.14}$$

where each terminal symbol is really a functor that will be applied to the input tokens by the parser.

Each of the functors are methods of a *WordPropertyChecker* object that has access to NorKompLeks. The functors take an input token as an argument. Therefore, each method can lookup the input token in NorKompLeks and perform arbitrary checks on it.

3.5.2 *The grammar*

This section will present the grammar used by the compound-word parser. The parser is responsible for following the guidelines suggested by Johannessen and Hauglin (1996) not covered by the algorithms presented above.

The grammar has a start symbol Σ which can expand to either a word or part of a word:

$$\Sigma \rightarrow \langle \text{word} \rangle \mid \langle \text{part of word} \rangle \tag{3.15}$$

The nonterminal $\langle \text{word} \rangle$ has the following productions:

Handling of Norwegian Compounds

$$\begin{aligned}
\langle \text{word} \rangle \rightarrow & \langle \text{noun} \rangle \\
\mid & \langle \text{verb} \rangle \\
\mid & \langle \text{adverb} \rangle \\
\mid & \langle \text{adjective} \rangle \\
\mid & \langle \text{preposition} \rangle \\
\mid & \langle \text{stem} \rangle \mid \langle \text{monosyllabic stem} \rangle \\
\mid & \langle \text{compound word} \rangle \mid \langle \text{compound word} \rangle \ \langle \text{genitive suffix} \rangle \\
\mid & \langle \text{noun} \rangle \ \langle \text{genitive suffix} \rangle \\
\mid & \langle \text{compound noun} \rangle \ \langle \text{genitive suffix} \rangle \\
\mid & \langle \text{prefix} \rangle \ \langle \text{prefix and verb joint} \rangle \ \langle \text{verb suffix} \rangle
\end{aligned} \tag{3.16}$$

which is what the function Get-Complete-Word-Parses, presented in Algorithm 3.4, looks for. While the following productions:

$$\begin{aligned}
\langle \text{part of word} \rangle \rightarrow & \langle \text{noun} \rangle \mid \langle \text{noun ing} \rangle \mid \langle \text{noun suffix} \rangle \\
\mid & \langle \text{verb} \rangle \\
\mid & \langle \text{adverb} \rangle \\
\mid & \langle \text{adjective} \rangle \mid \langle \text{adjective suffix} \rangle \\
\mid & \langle \text{preposition} \rangle \\
\mid & \langle \text{stem} \rangle \mid \langle \text{left stem} \rangle \mid \langle \text{right stem} \rangle \mid \langle \text{compound stem} \rangle \\
\mid & \langle \text{inflection formative} \rangle \mid \langle \text{derivation formative} \rangle \\
\mid & \langle \text{vowels joint} \rangle \\
\mid & \langle \text{prefix and verb joint} \rangle \\
\mid & \langle \text{prefix} \rangle \\
\mid & \langle \text{epenthetic e} \rangle \mid \langle \text{epenthetic s} \rangle \\
\mid & \langle \text{genitive suffix} \rangle
\end{aligned} \tag{3.17}$$

comprise the $\langle \text{part of word} \rangle$, which is what the function Get-Part-of-Word-Parses, presented in Algorithm 3.5, is looking for.

There are two important things to note about (3.16) and (3.17). The first thing is that $\langle \text{part of word} \rangle$ is not part of $\langle \text{word} \rangle$. The second thing is that the set of all nonterminal symbols on the right-hand side of $\langle \text{word} \rangle$ is not a subset of the set of all symbols on the right-hand side of $\langle \text{part of word} \rangle$, nor vice versa.

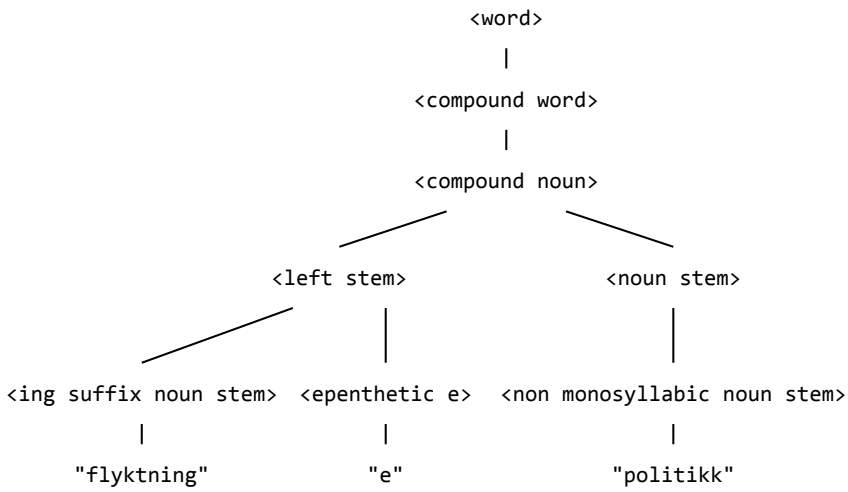


FIGURE 3.3: The highest prioritized analysis of *«flyktningepolitikk»* returned by the compound-word analyzer.

The reason for this is that ⟨word⟩ and ⟨part of word⟩ are used by different functions, and that a segmentation that is recognized by the first function—GET-PART-OF-WORD-PARSES—as a *part of a word* will be used as input *together with other segmentations* for the other function—GET-COMPLETE-WORD-PARSES. The interaction between algorithms 3.3 and 3.4 in Algorithm 3.2 should be studied carefully to see how this is done.

Johannessen (2001) claims that only stems of words can be compounded; not arbitrary inflected forms of words. The grammar described here is designed to adhere to that claim where practical, but with a few exceptions. For example, (JH35) states that an "epenthetic -e- can only be attached to a stem that is monosyllabic." However, enforcing that principle would inhibit a successful analysis of compounds like *«flyktningepolitikk»* ("refugee politics") and *«vikingetiden»* ("the Viking Period")[5] where the left stems are plurisyllables; see figures 3.3 and 3.4,

5 It should be noted that *Bokmålsordboka* (Wangensteen 2005) contains only the form *«vikingtiden»* (without the -e-). However, out of 1326 articles in the newspaper Aftenposten, 335 use only *«vikingetiden»*, 966 articles use only *«vikingtiden»*, and 25 articles use both forms.

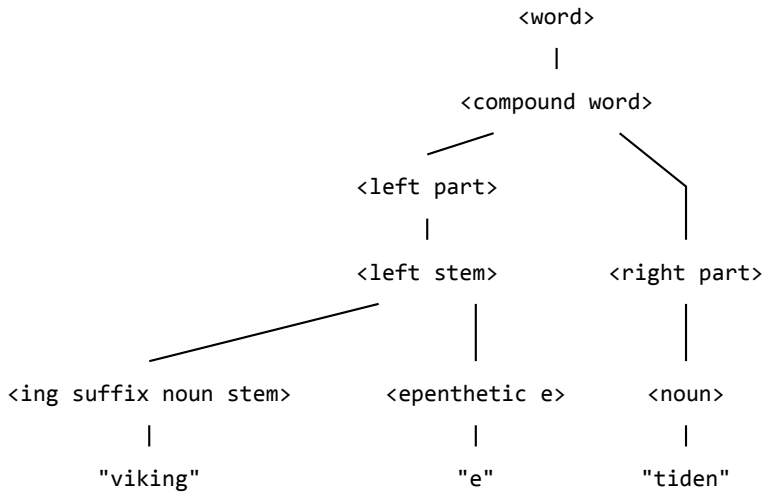


FIGURE 3.4: The highest prioritized analysis of «*vikingetiden*» returned by the compound-word analyzer.

respectively. Therefore an additional production was added to allow epenthetic -e- to attach to a noun-stem if it ends with "-ing":

$$\begin{aligned}\langle\text{left stem}\rangle \rightarrow &\langle\text{monosyllabic noun stem}\rangle\ \langle\text{epenthetic e}\rangle \\ \mid &\langle\text{ing suffix noun stem}\rangle\ \langle\text{epenthetic e}\rangle \\ \mid &\langle\text{monosyllabic verb stem}\rangle\ \langle\text{epenthetic e}\rangle\end{aligned} \tag{3.18}$$

It should be noted that these productions also adhere to (JH37). Furthermore, note that a stem with an attached epenthetic -e-, or -s-, is still considered a stem.

The following grammar rules are used to implement guidelines (JH22), (JH40), and (JH42):

$$\begin{aligned}\langle\text{left stem}\rangle &\rightarrow \langle\text{non sibilant ending noun stem}\rangle\ \langle\text{epenthetic s}\rangle \\ \langle\text{compound stem}\rangle &\rightarrow \langle\text{compound noun}\rangle\ \langle\text{epenthetic s}\rangle \\ &\mid \langle\text{may be compound noun}\rangle\ \langle\text{epenthetic s}\rangle\end{aligned} \tag{3.19}$$

Compound-Word Parser

As mentioned above, there is a strong focus on stems in (Johannessen 2001). This is also reflected in the implemented grammar:

$$
\begin{aligned}
\langle \text{compound stem} \rangle &\rightarrow \langle \text{left stem} \rangle \ \langle \text{right stem} \rangle \\
\langle \text{left stem} \rangle &\rightarrow \langle \text{stem} \rangle \mid \langle \text{monosyllabic stem} \rangle \mid \langle \text{compound stem} \rangle \\
&\quad \mid \langle \text{prefix} \rangle \mid \langle \text{quantifier} \rangle \\
\langle \text{right stem} \rangle &\rightarrow \langle \text{stem} \rangle \\
&\quad \mid \langle \text{monosyllabic stem} \rangle
\end{aligned}
$$

$$
\begin{aligned}
\langle \text{stem} \rangle &\rightarrow \langle \text{non monosyllabic noun stem} \rangle \\
&\quad \mid \langle \text{non monosyllabic verb stem} \rangle \\
\langle \text{monosyllabic stem} \rangle &\rightarrow \langle \text{monosyllabic noun stem} \rangle \\
&\quad \mid \langle \text{monosyllabic verb stem} \rangle \\
\langle \text{noun stem} \rangle &\rightarrow \langle \text{monosyllabic noun stem} \rangle \\
&\quad \mid \langle \text{non monosyllabic noun stem} \rangle \\
\langle \text{verb stem} \rangle &\rightarrow \langle \text{non monosyllabic verb stem} \rangle \\
&\quad \mid \langle \text{monosyllabic verb stem} \rangle \\
&\quad \mid \langle \text{verb stem vowel ending} \rangle
\end{aligned}
\tag{3.20}
$$

However, since Johannessen defines the stem of a word to be the part that does not change during inflection, some problems arise if the grammar should allow only stems to form compounds. Even though she also points out that the compound as a whole could be inflected according to the lexical category of the head of the compound, a problem arises with compounds, like *«raskestvoksende»* (“quickest growing”), where it seems that the left member is indeed an inflected adjective (*«rask»*,

"quick"). Therefore, some less strict productions are included in the grammar:

$$
\begin{aligned}
\langle\text{compound noun}\rangle &\rightarrow \langle\text{left stem}\rangle\ \langle\text{noun stem}\rangle \\
&\mid \langle\text{left stem}\rangle\ \langle\text{noun suffix}\rangle \\
&\mid \langle\text{verb stem}\rangle\ \langle\text{noun ing}\rangle \\
&\mid \langle\text{compound verb}\rangle\ \langle\text{noun ing}\rangle \\
&\mid \langle\text{verb stem vowel ending}\rangle\ \langle\text{vowels joint}\rangle\ \langle\text{noun suffix}\rangle \\
&\mid \langle\text{verb}\rangle\ \langle\text{preposition}\rangle\ \langle\text{noun}\rangle \\
&\mid \langle\text{adjective}\rangle\ \langle\text{noun}\rangle \\
&\mid \langle\text{quantifier}\rangle\ \langle\text{noun}\rangle \\
\langle\text{compound verb}\rangle &\rightarrow \langle\text{noun}\rangle\ \langle\text{verb stem}\rangle \\
&\mid \langle\text{adjective}\rangle\ \langle\text{verb stem}\rangle \\
&\mid \langle\text{adverb}\rangle\ \langle\text{verb stem}\rangle \\
\langle\text{compound word}\rangle &\rightarrow \langle\text{compound noun}\rangle \\
&\mid \langle\text{left part}\rangle\ \langle\text{right part}\rangle \\
&\mid \langle\text{left part}\rangle\ \langle\text{compound noun}\rangle \\
&\mid \langle\text{noun stem}\rangle\ \langle\text{adjective suffix}\rangle \\
\langle\text{left part}\rangle &\rightarrow \langle\text{left stem}\rangle \mid \langle\text{noun}\rangle \mid \langle\text{compound noun}\rangle \\
&\mid \langle\text{adjective}\rangle \mid \langle\text{adverb}\rangle \\
\langle\text{right part}\rangle &\rightarrow \langle\text{noun}\rangle \mid \langle\text{adjective}\rangle \mid \langle\text{verb}\rangle \\
\langle\text{adjective}\rangle &\rightarrow \langle\text{compound noun}\rangle\ \langle\text{derivation formative}\rangle \\
&\mid \langle\text{adjective}\rangle\ \langle\text{inflection formative}\rangle
\end{aligned}
\tag{3.21}
$$

As can be seen from Figure 3.5, this enables the analyzer to succeed with its analysis of *«raskestvoksende»*.

Compound-Word Parser

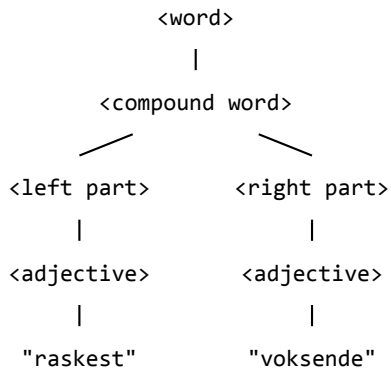


FIGURE 3.5: The highest prioritized analysis of *«raskestvoksende»* returned by the compound-word analyzer.

Finally, there are the grammar rules containing the terminal symbols; each one a functor that checks for what its name suggests:

$$\begin{aligned}
\langle \text{adjective} \rangle &\rightarrow \textsc{adjective?} \\
\langle \text{adjective suffix} \rangle &\rightarrow \textsc{adjective-suffix?} \\
\langle \text{adverb} \rangle &\rightarrow \textsc{adverb?} \\
\langle \text{prefix and verb joint} \rangle &\rightarrow \textsc{prefix-verb-joint?} \\
\langle \text{vowels joint} \rangle &\rightarrow \textsc{vowels-joint?} \\
\langle \text{derivation formative} \rangle &\rightarrow \textsc{derivation-formative?} \\
\langle \text{epenthetic e} \rangle &\rightarrow \textsc{epenthetic-e?} \\
\langle \text{epenthetic s} \rangle &\rightarrow \textsc{epenthetic-s?} \\
\langle \text{genitive suffix} \rangle &\rightarrow \textsc{genitive-suffix?} \\
\langle \text{inflection formative} \rangle &\rightarrow \textsc{inflection-formative?} \\
\langle \text{ing suffix noun stem} \rangle &\rightarrow \textsc{ing-suffix-noun-stem?} \\
\langle \text{may be compound noun} \rangle &\rightarrow \textsc{may-be-compound-noun?} \\
\langle \text{monosyllabic noun stem} \rangle &\rightarrow \textsc{monosyllabic-noun-stem?} \\
\langle \text{monosyllabic verb stem} \rangle &\rightarrow \textsc{monosyllabic-verb-stem?} \\
\langle \text{non monosyllabic noun stem} \rangle &\rightarrow \textsc{non-monosyllabic-noun-stem?} \\
\langle \text{non monosyllabic verb stem} \rangle &\rightarrow \textsc{non-monosyllabic-verb-stem?}
\end{aligned} \tag{3.22}$$

⟨non sibilant ending noun stem⟩ →NON-SIBILANT-ENDING-NOUN?
⟨noun⟩ →NOUN?
⟨noun ing⟩ →NOUN-ING?
⟨noun stem⟩ →NOUN-STEM?
⟨noun suffix⟩ →NOUN-SUFFIX?
⟨prefix⟩ →PREFIX?
⟨preposition⟩ →PREPOSITION?
⟨quantifier⟩ →QUANTIFIER?
⟨verb⟩ →VERB?
⟨verb stem vowel ending⟩ →VERB-STEM-VOWEL-ENDING?
⟨verb suffix⟩ →VERB-SUFFIX?

(3.23)

A few of the rules here are rather pragmatically motivated. For example, the noun «*tiltredelse*» ("accession") is not really a compound, nor is it covered by NorKompLeks. Furthermore, for many verbs, a noun can be derived from it by adding the ending «*-else*», which is an entry in NorKompLeks. In this case, it seems plausible that the noun was derived from the verb «*tiltre*» ("enter into") by adding the «*-else*» ending. However, it seems that if the verb ends with an 'e', a "joint" character is needed. In this case, the joint should be a 'd'. Unfortunately, «*-delse*» is not an NorKompLeks entry. Therefore, the ⟨vowels joint⟩ production checks whether the input token equals `"d"`, to handle such words.

3.6 Results

To gain an idea of how well the automatic compound-word analyzer—which is part of the NorKompLeks module—was able to handle previously unseen compounds, the analyzer was first tested separately.

Since the analyzer would be integrated into the mapping framework that will be presented in Section 4.9, these results also give some indications of how well the mapping framework will handle compounds.

To test the analyzer, a corpus comprising five recent, arbitrarily chosen articles from the Norwegian national newspaper Aftenposten was used; see Table 3.2 on the next page. As shown in the table, the total size of the corpus was 4,951 words, and it contained 109 compound words not found in the NorKompLeks lexicon. All those compounds constituted unseen data that the compound-word analyzer never had been tested with.

Table 3.2: Test results for the automatic compound-word analyzer applied to compounds found in newspaper articles.

Results

		Compound analyzes		
Text	Words	Correct	Incorrect	Success rate (%)
Newspaper article[a]	1,420	21	0	100
Newspaper article[b]	652	18	0	100
Newspaper article[c]	472	25	0	100
Newspaper article[d]	1,238	17	0	100
Newspaper article[e]	1,169	28	0	100
Total	4,951	109	0	100

a Ingrid Brekke, «Andre omstridte kulturskatter,» *Aftenposten Morgen*, sec. 2, December 23, 2007.
b Gunhild M. Haugnes, «Halvparten har fri programvare: Microsoft får stadig tøffere konkurranse,» *Aftenposten Morgen*, sec. 3, December 18, 2007.
c NTB, «Nettbutikken iTunes åpnet i natt,» *Aftenposten Aften*, sec. 1, May 10, 2005.
d Lars-Ludvig Røed, «Soldater blir overgripere i krig,» *Aftenposten Morgen*, sec. 1, September 27, 2007.
e Paul C. Taylor, «Norge i et nøtteskall,» *Aftenposten Morgen*, sec. 2, December 24, 2007.

Furthermore, Table 3.2 also shows the results of the test. Since the compound-word analyzer actually returns a prioritized list of theoretically valid analyses, an analysis was defined to be correct if the analysis with the highest priority was both correctly split up and every part of the compound was assigned a correct lexical category. For this test, the examination of the results was done by the author. As the table shows, all the compounds were analyzed correctly.

However, the analyzer was also tested on 48 example compounds found in the article about automatic compound analysis by Johannessen and Hauglin (1996). Of the examples, 42 compounds were both correctly analyzed and ranked, while 6 results returned by the analyzer contained the correct analysis, but the correct analysis was incorrectly not given the highest priority. This corresponds to a success rate of 87.5 %.

Of the 6 unsuccessful results, 3 were caused by the analyzer ranking an analysis with otherwise identical members as the correct analysis, except for the head belonging to a wrong lexical category, highest. For those results, the analysis with the *correctly* categorized head was given the second highest priority. The final 3 unsuccessful results were all ambiguous epenthetic -s- cases where the correct analysis was found

among the top three candidates, but the highest-ranked analyses were wrong.

3.7 Analysis and Discussion

As shown in Section 3.6, the compound-word analyzer performed perfectly on previously unseen compounds from a random sample of newspaper articles.

However, when testing the analyzer on the example compound words discussed by Johannessen and Hauglin (1996), the analyzer made a few errors during ranking of the analyses. In three of the six ranking-error cases, the correct analyses were ranked as the second most probable analysis, while in the other three cases, the correct analyses were ranked as the third most probable analysis.

Because the few errors that occurred were caused by incorrect *ranking* of the analyses—while the correct analyses were among the top three candidates—it seems safe to conclude that the grammar and analysis algorithms are adequate for the task.

On the other hand, *ranking* the analyses correctly has proved to be more difficult. At one point during the development of the analyzer, all the example compounds discussed by Johannessen and Hauglin (1996) were correctly both analyzed *and* ranked. However, that ranking scheme produced erroneous rankings when applied to previously unseen real-world texts from, for example, newspaper articles. Furthermore, several attempts at defining rules that would correctly rank both the examples from the article and real-world examples did not succeed.

The overall performance of the compound-word analyzer was nonetheless satisfying, and showed that the analyzer was adequate for use in Verto.

For comparison, Kokkinakis et al. (2000) reported that their algorithm using an *n*-gram-based heuristic for Swedish compound segmentation achieved over 95 % precision. Furthermore, Pedersen (2007) reported that the compound-word analyzer used in her research for splitting Danish compounds had an error rate of less than 1 %.

3.7.1 *Comparison with Johannessen and Hauglin's Compound Analyzer*

The compound analyzer described in the article by Johannessen and Hauglin (1996) constituted a module in the Oslo-Bergen tagger being developed at the time. According to the Text Laboratory (2008a), the compound analyzer was originally developed and implemented at the Text Laboratory at the University of Oslo (UIO), but has later been

reimplemented by *Paul Meurer* at the *Centre of Culture, Language and Information Technology* (Aksis) at the University of Bergen (UIB).

Analysis and Discussion

In their article, Johannessen and Hauglin present some preliminary evaluation results. They state that to succeed, their analyzer must accomplish two tasks; it must be able to "find all analyses of all compounds in a text" (segmentation and parsing), and to "find the correct one amongst several possibilities" (ordering, or ranking). Furthermore, they report that their analyzer made a wrong analysis in 1.1 % of the cases, and a partly wrong analysis in 1.3 % of the cases. They count results where the analysis with the highest assigned priority has a wrong head as wrong analyses. As a partly wrong analysis, they count analyses where the final member is correct on its own, but where incorrect segmentation gives the compound as a whole an incorrect meaning.

According to the criteria used in Section 3.6, both of the two kinds of errors they describe would be counted as errors. According to these criteria, their analyzer had a success rate of $100\ \% - (1.3\ \% + 1.1\ \%) = 97.6\ \%$.

However, they do not describe their test data at all; neither the source texts nor the number of compounds tested. Hence, it is not certain that a comparison of success rates may be valid. If the comparison should be made, it is hard to decide which of the success rates from Section 3.6 to compare with; the 100 % success rate on arbitrarily selected newspapers, or the 87.5 % success rate on compounds used as examples in Johannessen and Hauglin's article—some of which might be considered a bit strained. Perhaps the most representative success rate for the compound-word analyzer presented herein would be one based on the average of the two tests, that is

$$\frac{\text{total \# of correct analyses}}{\text{total \# of compounds}} = \frac{109 + 42}{109 + 48} = 96.2\ \%. \tag{3.24}$$

Nonetheless, on the average the success rate compares satisfactory with Johannessen and Hauglin's results, and herein the test data has been described.

A small, but important, detail in the implementation of the compound-word analyzer resulted in an improvement over the compound-word analyzer from the Oslo-Bergen tagger[6]. As shown in (3.8)–(3.11), Algorithm 3.7 handles the omission of redundant identical consonants when compounding Norwegian words. However, trying to analyze the same kinds of compounds with the compound-word analyzer from the

6 Available on-line at `http://decentius.hit.uib.no:8005/cl/cgp/ranked-analyses.xml` (last visited January 29, 2008).

TABLE 3.3: The ranked analyses returned by the Oslo-Bergen tagger's compound-word analyzer when analyzing «*musikkatalog*» ("music catalog").

Analysis #	Component	Lemma	Feature
0	musik	musik	ukjent
	katalog	katalog	subst appell mask ub ent
1	musikkata	musikkata	ukjent
	log	log	subst appell mask ub ent
2	musikkata	musikkata	ukjent
	log	log	subst appell mask ub ent
3	mu	mu	ukjent
	sik	sik	subst appell fem ub ent
	katalog	katalog	subst appell mask ub ent
4	mu	mu	ukjent
	sik	sik	subst appell nøyt ub ent
	katalog	katalog	subst appell mask ub ent
5	mu	mu	ukjent
	sik	sik	subst appell mask ub ent
	katalog	katalog	subst appell mask ub ent

Oslo-Bergen tagger discloses that the analyzer does not handle them correctly. For example, the ranked analyses returned by the Oslo-Bergen tagger's analyzer is shown in Table 3.3 on the preceding page. It should be noted that every one of the returned analyses has a first member, or component, that is *«ukjent»* ("unknown") to the analyzer because it does not compensate for the consonant omission. On the other hand, in the highest ranked analysis, the head of the compound is correct. Correctly identifying the head of the compound can be enough for some uses, but since each member of the analyses returned by the compound-word analyzer in Verto are used for mapping, *all* the members must be correct.

3.7.2 *Efficiency*

Another difference between Johannessen and Hauglin (1996)'s compound-word analyzer and the analyzer presented herein is that their analyzer was implemented with regular expressions (regexps), while my analyzer was implemented with the use of a CFG, as detailed in Section 3.5.

The reason for using a CFG-based approach was twofold. One aspect was that I found it easier both to develop, experiment with, and maintain the analyzer using Context-Free Grammars (CFGs) instead of regexps. The other aspect was that a CFG-based approach made it less probable to end up in a situation where the generative power of the grammar formalism would inhibit the development of the analyzer.

However, if the system ought to be used in, for example, a commercial setting where speed matters more, I would suggest rewriting the compound-word analyzer to use regexps instead of a CFG. Doing so would greatly improve the speed of the analyzer, because every regular expression (regexp) can be represented by a finite-state automaton (FSA), or finite-state transducer (FST), and an FSA can recognize an input string of length n in $O(n)$ time (Lewis and Papadimitriou 1998; Hamburger and Richards 2002).

Section 6.1 discusses how selecting the correct compound analysis can be improved by exploiting the semantic resource generated by Verto.

Mapping Norwegian to WordNet

4

This chapter presents the method I developed for automatically mapping content words—covering nouns, verbs, and modifiers like adjectives and adverbs—in a non-English source language to WordNet senses. The method was implemented as a computer program named Verto. Hence, Verto represents the concretization of the method.

In the following presentation of the method, the specific source language for the mapping process is *Norwegian*. However, there are no compelling reasons that the method should not be applicable to mapping words from other languages to concepts in WordNet as well.

4.1 Resources

The basic principle of the method for automatically mapping Norwegian content words to concepts in WordNet—that will be presented in detail from Section 4.2 onwards—depends on the availability of the following three lexical resources:

1 A dictionary containing translations from the source language to the target language;

2 A dictionary containing translations from the target language to the source language; and

3 The target ontology of the mapping, a lexical-semantic resource for the target language.

To be able to handle mapping of compound words, Verto incorporates the compound-word analyzer described in Chapter 3 as a module. Through this module, another dependency can be added to the above list:

4 A lexicon containing morphological information about Norwegian words.

Resources

skatt *subst. m* **1** treasure
2 tax
3 darling, dear, sweetheart, love *(britisk)*, honey *(amer.)*
betale skatt pay tax(es)
direkte skatt direct tax
få igjen penger på skatten get a tax refund
i skatt in taxes
legge skatt på noe impose a tax on something
proporsjonal skatt proportional tax, flat tax
skatten min my dear, my darling, sweetheart
snyte på skatten cheat on (one's) taxes, evade tax
trekke skatt deduct tax(es)
unndra skatt evade tax
utsatt skatt deferred tax
⋮
skatte *verb* **1** pay tax(es)
2 *(skattlegge)* tax, assess
3 *(sette pris på)* appreciate
skatte av inntekt tax one's income
være høyt skattet be greatly appreciated

FIGURE 4.1: The entries for the Norwegian words «*skatt*» (noun) and «*skatte*» (verb) in *Norsk–engelsk stor ordbok* (Haslerud and Henriksen 2003).

The latter resource is provided by the NORKOMPLEKS-module already described in Section 3.2.

Although the method developed should be applicable to other languages, the following discussion will consider the source language to be Norwegian and the target language to be English.

Table 4.1 on the next page shows a brief quantitative overview of the three lexical resources used, *Norsk–engelsk stor ordbok*, *Engelsk–norsk stor ordbok*, and *Norsk komputasjonelt leksikon* (NorKompLeks). It should be noted that the method presented is only concerned with mapping of content words; that is, nouns, adjectives, verbs, and adverbs.

Table 4.1: Number of keywords and translations (where applicable) in the lexical resources NorKompLeks, *Norsk–engelsk stor ordbok*, *Engelsk–norsk stor ordbok*.

Lexical category	NorKompLeks	*Norsk–engelsk stor ordbok*		*Engelsk–norsk stor ordbok*	
	Keywords	Keywords	Translations	Keywords	Translations
Noun	51,655	45,715	137,435	54,918	203,583
Adjective	19,216	9,241	36,270	16,204	69,519
Verb	6,931	5,453	38,336	8,840	93,420
Adverb	1,084	635	3,704	1,659	8,705
Preposition	235	174	975	133	1,365
Interjection	212	124	474	349	916
Determiner	64	106	880	103	1,064
Conjunction	8	38	207	49	310
Subjunction	37	5	38	18	175
Pronoun	33	26	171	66	754

Next, each of the former three resources mentioned above will be presented in more detail.

4.1.1 Human/Machine-Readable Dictionaries

Resources

The Norwegian–English and English–Norwegian dictionaries made available for this research were *Norsk–engelsk stor ordbok* and *Engelsk–norsk stor ordbok*, comprising the bilingual dictionary *Engelsk stor ordbok: Engelsk-norsk / norsk-engelsk* (Haslerud and Henriksen 2003). Figure 4.1 on page 72 shows a couple of example entries in *Norsk–engelsk stor ordbok*, with keywords typeset in bold, followed by their lexical category and gender (for nouns) typeset in italics; the rest of each entry follows slightly indented.

All senses are enumerated in bold. Furthermore, for *some* of the senses, several suggested translations are listed, separated by commas. Below the enumerated senses, a list of example expressions in Norwegian are listed along with their translations.

The first example in Figure 4.1 is for the noun *«skatt»* and the second for the verb *«skatte»*. As can be seen from the figure, both the noun *«skatt»* and the verb *«skatte»* has three senses, while all but two example expressions concern the kind of tax usually imposed by authority. Hence, there is no guarantee of a balanced mix of examples of usage between the senses. Furthermore, the expressions do not indicate which sense they concern.

Also shown in the figure is the different information provided in parentheses. After the sense numbers, a parenthesis indicates what sense it corresponds to in Norwegian. A parenthesis placed after a suggested translation expresses information about differing usage in, for example, British, Canadian, and American English. Such parenthetical remarks can also indicate slang translations. Other places, parentheses are used for indicating the plural form of words, or expressing words indicative of the translation's usage.

The publisher of *Norsk–engelsk stor ordbok* provided the sources for the printed bilingual dictionaries, formatted as Extensible Markup Language (XML). As such, they constituted machine-readable dictionaries.

There is no reason to believe that the XML format is inappropriate for typesetting the dictionaries, but it is hardly a format that inherently provides fast, automatic lookups. Furthermore, since a central part of the method is based on looking up words in these dictionaries, the need for some kind of efficient interface was apparent.

Since no document type definition (DTD), or other description of the format, accompanied the files, quite a lot of work was put into

parsing and restructuring the dictionaries. A lot of small structural inconsistencies—most of them invisible to readers of the printed dictionary, but never ignored by a pedantic parser—led to quite a few suggested corrections.[1] In the end, the XML documents were parsed and the adequate information extracted and converted into a tabular format.

Mapping Norwegian to WordNet

One should note that all but one of the different kinds of parenthetical remarks—like the ones shown in Figure 4.1 on page 72—are tagged in the XML files, indicating the nature of the remark. The only kinds of parenthetical remark not tagged are the ones mostly used for describing plural forms and optional words. These parentheses, along with slashes mostly indicating interchangeable words, resulted in multiple entries in the tabular format; one for each possible form.

During the transformation into tabular form, an effort was made to conserve as much relevant information per entry as possible. That information includes the lexical class and gender, whether something is only used in a variety of English (like American, and Canadian), and whether it is a translation that contains an example-phrase (in the following called phrasal translation).

The tabular files representing the dictionaries were then transformed into compact data objects, based on the trie data structure. Small program modules that when given a keyword returned a list of associated information then wrapped these objects.

Each list item consists of a tuple representing a possible translation. The tuple consists of the suggested translation, a set of attributes, and a number identifying the sense of the keyword in the source language. For example, looking up the Norwegian word *«datamaskin»* in the *Norsk–engelsk stor ordbok* program module yields

$$\begin{aligned}\textsc{NorEng}(\text{datamaskin}) = \\ \langle &(\text{computer}, \{(\text{gend}, \text{m}), (\text{pos}, \text{subst})\}, 8088) \\ &(\text{laptop (bærbar datamaskin)}, \\ &\quad \{(\text{gend}, \text{m}), (\text{phrase}, \text{true}), (\text{pos}, \text{subst})\}, 8088) \\ &(\text{personal computer (personlig datamaskin)}, \\ &\quad \{(\text{gend}, \text{m}), (\text{phrase}, \text{true}), (\text{pos}, \text{subst})\}, 8088)\rangle.\end{aligned} \tag{4.1}$$

where NorEng designates the *Norsk–engelsk stor ordbok* module.

As (4.1) shows, if a suggested translation constitutes an expression, this is indicated by the attribute pair $(\text{phrase}, \text{true})$ and a parenthesis

1 *Martin Thorsen Ranang*, e-mail message to *Vibecke C. D. Haslerud*, March 19, 2004.

smoothie[1] *subst.* /ˈsmuːðɪ/ eller **smoothy** *(hverdagslig)* sleiping, slesk person, innsmigrende person
smoothie[2] *subst.* /ˈsmuːðɪ/ smoothie *(leskende næringsdrikk med frukt, bær og youghurt som basisingredienser)*

FIGURE 4.2: The entries for the English noun "smoothie" in *Engelsk–norsk stor ordbok*. (Source: Haslerud and Henriksen (2003).)

containing the Norwegian expression following the English translation. The presented method does not use the gender information.

The *Engelsk–norsk stor ordbok* module, referred to as ENGNOR, returns the same kind of information, except that no gender is specified, because the English language lacks grammatical gender (except, for example, on pronouns) (Trask 1993; p. 115). For example, Figure 4.2 shows the entries in *Engelsk–norsk stor ordbok* for the English word "smoothie", and a lookup of "smoothie" in the corresponding program module results in the list

$$\begin{aligned}&\textsc{EngNor}(\text{smoothie}) = \\ &\quad\langle(\text{sleiping}, \{(\text{pos}, \text{subst})\}, 67158) \\ &\quad\ (\text{slesk person}, \{(\text{pos}, \text{subst})\}, 67158) \\ &\quad\ (\text{innsmigrende person}, \{(\text{pos}, \text{subst})\}, 67158) \\ &\quad\ (\text{smoothie}, \{(\text{pos}, \text{subst})\}, 67159)\rangle.\end{aligned} \tag{4.2}$$

As the last element of each tuple in this example shows, the first three tuples returned all refer to the first sense of the keyword while the last tuple refers to the second sense.

4.1.2 WordNet

The target of the mapping performed by the method implemented in Verto is the WordNet ontology (Miller et al. 1990; Fellbaum 1998c), version 2.1 (Miller and Hristea 2006). No changes were made to it.

Over twenty years ago, Miller (1985) wrote that

> Although there is no principled reason why natural language processors should not have vocabularies large enough

> to deal with a [*sic*] any domain of topics, we are presently far from having such vocabularies on line.

To improve on that situation, he and others have since developed WordNet, an electronic lexical database for English. And, perhaps most importantly, they did this—as Fellbaum (1998c; p. 137) states—"not just for a handful of words but for the better part of the vocabulary of a natural language."

Mapping Norwegian to WordNet

An important design criterion during the development of WordNet has been to organize it according to findings in "psycholinguistic research on the lexical component of language" (Miller 1985). Consequently, WordNet constitutes a semantic network where the semantic relationships that hold between different word senses are easily accessible. WordNet quickly became the *de facto* semantic network in Natural Language Processing (NLP) research (Fellbaum 1998a; Jurafsky and Martin 2000).

A concept in the WordNet ontology is defined as a synonym set (synset). That is, a word in WordNet may have multiple senses, and each sense of a particular word belongs to a specific synset. Furthermore, all the senses in a particular synset are synonymous. Hence, each sense of a word belongs to one and only one synset, and each synset represents one semantic concept, or meaning.

It should be noted that I used the word "word" quite liberally in the previous paragraph. WordNet does not discriminate between simplex words (for example, "cat", or "engine"), compound words (for example, "letter box", or "coffee cup"), or idioms (for example, "kick the bucket", or "red carpet"); they are all represented as strings. Note also that the idiom "kick the bucket" has only *one* sense in WordNet. Any other interpretation of that phrase must be composed by the semantics of its parts.

Each concept, or synset, is accompanied by a definitional gloss. For example, the three senses of "engine" in WordNet are paired with the glosses shown in Table 4.2. Some of the glosses also contain example sentences, as shown in the gloss for {`engine_n_2`}.

Additionally, verb frames that describe selectional restrictions for the verb accompany many of the verbs. For example, one of the frames for `design_v_2` (design something for a specific role or purpose or effect; "This room is not designed for work")[2] is "Somebody —s something", where —s is a place holder for the verb, which indicates that the agent of the design action must be animate, while the patient must be an object.

2 The gloss from WordNet.

TABLE 4.2: WordNet glosses for the three synsets representing the different senses of engine.

Synset	Gloss
{`engine_n_1`}	motor that converts thermal energy to mechanical work
{`engine_n_2`}	something used to achieve a purpose; "an engine of change"
{`locomotive_n_1`, `engine_n_3`, `locomotive_engine_n_1`, `railway_locomotive_n_1`}	a wheeled vehicle consisting of a self-propelled engine that is used to draw trains along railway tracks

TABLE 4.3: Unique strings, synsets, and word senses in WordNet.

Lexical category	Unique strings[a]	Synsets	Word-sense pairs
Noun	117,097	81,426	145,104
Adjective	22,141	18,877	31,302
Verb	11,488	13,650	24,890
Adverb	4,601	3,644	5,720
Total	155,327	117,597	207,016

a The sense of unique here is *per lexical category*. The total number of completely unique strings, ignoring the lexical categories, is 147,249.

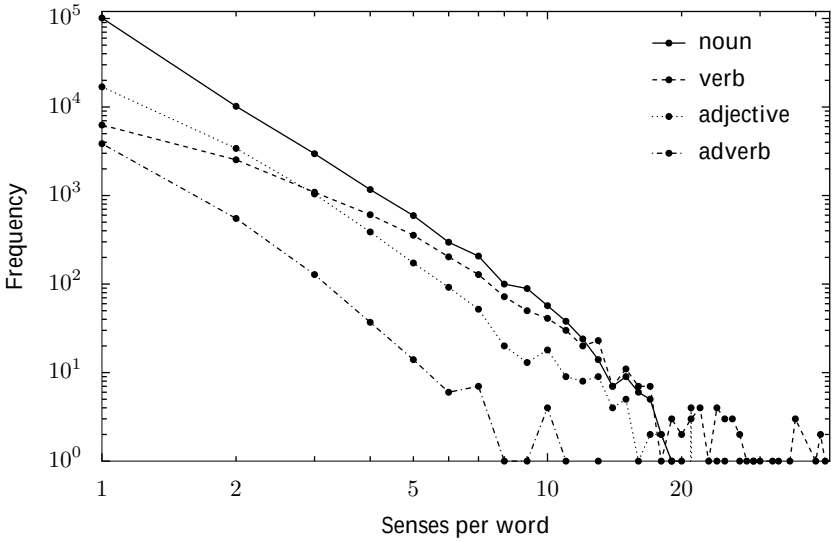


FIGURE 4.3: Polysemy count for each lexical category in WordNet. It should be noted the use of logarithmic scales on both axes. It should also be noted that the plot is generated from integer values; the connecting lines are provided only to show the trends, not to indicate continuity.

Table 4.3 shows the number of strings and synonym sets (synsets) WordNet contains information about. The third column reports the sum over all the senses for all the strings in WordNet. That is, if $\text{senses}(w, C)$ is a function that returns all the senses for a given word w belonging to the lexical category C, then each row in the third column equals

$$\sum_{w \in \text{WordNet}} |\text{senses}(w, C)| \tag{4.3}$$

for a given lexical category $C \in \{\text{noun}, \text{adjective}, \text{verb}, \text{adverb}\}$.

Figure 4.3 shows the degree of polysemy for each lexical category in WordNet. The figure shows that, for all the lexical categories, the frequency of words representing a given number of senses decreases radically as the number of senses increases. For example, in accord with Table 4.5 on page 83, the figure shows that 101,321 nouns are single sense words. However, combining this information with Table 4.3,

TABLE 4.4: Relations defined for different lexical categories in WordNet.

Relation	Lexical category			
	Noun	Verb	Adjective	Adverb
Synonymy	✓	✓	✓	✓
Antonym	✓	✓	✓	✓
Holonym/meronym	✓			
Instance hypernym/hyponym	✓			
Hypernym/hyponym[a]	✓	✓		
Entail		✓		
Cause		✓		
Group		✓		
Similar			✓	

a The "hyponyms"of verbs are called troponyms.

shows that only 15,776 nouns represent *two or more* senses. Furthermore, only 5,590 nouns represent *three or more* senses. The same trend applies to each of the other lexical categories.

Table 4.4 shows what relations are defined to hold between synsets within each lexical category. Of the relations shown in the table, the group—also referred to as verb group—relation should be explained further. The lexicographers use the group relation to manually group together some semantically related verbs. Hence, the verb group relation can be considered a manually defined similarity relation for verbs. Membership of a verb group is specified through ordered pairs of synsets (A, B) that indicate that A and B belong to the same group. Through transitivity, such two-element groups that share the same synsets are combined to form the largest groups possible. This means, for example, that if two minimal groups are defined by the pairs (A, B) and (B, C), then since the synset B is shared by both groups, they are combined to form a larger group $\{A, B, C\}$.

Limitations of WordNet

Even though much work has been put into enhancing WordNet both morphologically and semantically over the years (Harabagiu et al. 1999), WordNet has its limitations as an NLP resource.

One of the shortcomings WordNet has been criticized for is that it does not distinguish between hypernyms of classes and hypernyms of instances (Gangemi et al. 2001). However, version 2.1 of WordNet[3] introduced exactly this distinction (Miller and Hristea 2006). Because version 2.1 of WordNet is the target of the mapping performed by the method implemented in Verto, this enhancement has consequences even for the work presented herein, as shown in Section 4.3.

Another shortcoming of WordNet is that it lacks relations between topically related concepts. To alleviate that shortcoming, Agirre et al. (2000) enriched WordNet with topic signatures.

A limitation of WordNet that is often mentioned in the literature is that it does not encode thematic relations, or selectional restrictions, on nouns that function as arguments of specific verbs. However, this limitation is addressed by other, complementary semantic resources, such as the corpus-based frame-semantic resource FrameNet (Fillmore et al. 2003; Baker et al. 2003) and the verb lexicon "with explicitly stated syntactic and semantic information, using Levin verb classes to systematically construct lexical entries" VerbNet (Kipper et al. 2000a, 2004).

Regardless of the shortcomings and limitations of WordNet, it is still the *de facto* semantic resource in NLP. And, because of its popularity, hordes of researchers will continue to improve, enhance, and extend WordNet in the future. Some researchers also work on integrating WordNet with other complementary semantic resources, like Shi and Mihalcea (2005) who combine FrameNet, VerbNet, and WordNet for robust semantic parsing.

4.2 The Crux

The following sections will present the main problem of, and suggested solution to, the task of automatically mapping Norwegian words to those concepts in the WordNet ontology that express the semantics of the lexemes that the Norwegian words may represent.

In the following discourse, the assumption is made that the available resources are NorKompLeks, NorEng, EngNor, and WordNet, as described in the previous sections. Furthermore, the case is first considered for Norwegian simplex words; that is, not compound words and not sequences of words (neither collocations nor idioms).

A Norwegian word can represent multiple lexemes. For example, the word *«skatt»* may represent both the imperative form of a verb, with

3 WordNet 2.1 was released in March, 2005.

the morphemes {*«skatt»*, *«skatte»*, *«skatter»*, *«skattet»*}, and the nondefinite singular form of the noun with the morphemes {*«skatt»*, *«skatter»*, *«skatten»*, *«skattene»*}. Both the noun and the verb has multiple senses. The base form, or stem, of the verb is its infinitive form *«skatte»*, while for the noun it is the singular nondefinite form *«skatt»*. In most dictionaries, the keywords are base forms of words.

Handling of Instance Synonyms

When looking up a word—considering its lexical category (or part of speech (POS))—in the dictionary, a list of possible translations is found. The list may be empty, in the case that there is no defined translation. To continue the above example, the Norwegian noun *«skatt»* may be translated into the English nouns "tax", "treasure", and "honey".

Furthermore, each word in the target language may have one or more senses. For example, in WordNet the English noun "honey" has two different senses, one being "a sweet yellow liquid produced by bees", with the other being "a beloved person; used as terms of endearment." However, only the latter sense has a meaning that is warranted by the Norwegian noun *«skatt»*.

Thus, a single word in the source language may lead to a plethora of possible mappings to senses in the target language ontology. The core of the problem is then, how one can remove mappings to senses in the target language ontology that are not warranted by the semantics of the word in the source language.

The most naïve approach to performing the mapping from a Norwegian word to WordNet senses would be to keep all the suggested mappings. However, this has already been proved not to work through the above counterexample.

The following sections present a possible solution to the problem described above.

4.3 Handling of Instance Synonyms

The method developed herein—as a suggested solution—will not attempt to find mappings to senses representing instance synsets; that is, any synset in WordNet that is an instance hyponym of another (class) synset will be ignored. In other words, the method will not try to find mappings of proper names.

Table 4.5: Words per lexical category in WordNet that only represent a single sense each.

Lexical category	Words in WordNet	Single-sense words	% of words
Noun	117,097	88,643[a]	75.70
Adjective	22,141	16,889	76.28
Verb	11,488	6,261	54.50
Adverb	4,601	3,850	83.68

a This number does not include the 12,678 single-sense nouns that represent instances.

4.4 Single-sense Words

If an English word w, belonging to a particular lexical category `C`, only has *a single* sense defined in WordNet, represented by `w_C_1`, belonging to the synset S, so that

$$\begin{aligned} \text{synsets}(w, C) &= \langle S \rangle \\ &= \langle \{\texttt{w\_C\_1}, \texttt{x}_1\texttt{\_C\_i}_{\texttt{x}_1}, \texttt{x}_2\texttt{\_C\_i}_{\texttt{x}_2}, \dots\} \rangle , \end{aligned} \tag{4.4}$$

then the basic assumption made is that the suggested mapping is kept. Knight and Luk (1994) also made the same assumption.

A consequence of this assumption is that in such single-sense cases the complementary synonyms in the synset, represented by the set

$$S - \{\texttt{w\_C\_1}\} = \{\texttt{x}_j\texttt{\_C\_i}_{\texttt{x}_j} | j = 1, 2, \dots\} \tag{4.5}$$

do not affect the decision whether to accept the mapping to that sense or not.

By inspecting WordNet, we can get an exact overview of the number of words per category that have a single sense, as shown in Table 4.5.

4.5 Exploiting the Synonymy within Synsets

To emphasize an important point, the proposed method does *not* use any sense information present *in the dictionary resources*.[4] However, it does use the sense information contained in WordNet.

4 Even though the printed dictionary does discern between some senses, the implemented system ignores this information.

As mentioned in Section 4.1.2, the semantic relations in WordNet are defined to hold between synsets. In other words, each synset in WordNet represents a concept in the ontology that WordNet constitutes.

Furthermore, as each sense of a word belongs to only one synset, this means that all the different senses that constitute a particular synset

Exploiting the Synonymy within Synsets

$$S = \{\texttt{x}_1\texttt{\_C\_i}_{\texttt{x}_1}, \texttt{x}_2\texttt{\_C\_i}_{\texttt{x}_2}, \dots\} \tag{4.6}$$

all refer to *the same semantic concept*. In other words, each sense in the synset S is just another symbolic name for the same concept.

For example, in WordNet the different senses of the noun "chest" are represented by the synsets

$$\begin{aligned} \text{synsets}(\textit{chest}, n) &= \langle S_{\texttt{chest\_n\_1}}, S_{\texttt{chest\_n\_2}}, S_{\texttt{chest\_n\_3}}, \rangle \\ &= \langle \{\texttt{thorax\_n\_2}, \texttt{chest\_n\_1}, \texttt{pectus\_n\_1}\}, \\ &\quad \{\texttt{chest\_n\_2}\}, \\ &\quad \{\texttt{chest\_of\_drawers\_n\_1}, \texttt{chest\_n\_3}, \\ &\quad\ \texttt{bureau\_n\_2}, \texttt{dresser\_n\_2}\}\rangle. \end{aligned} \tag{4.7}$$

Consequently, `thorax_n_2`, `chest_n_1`, and `pectus_n_1` all refer to the same semantic concept ("the part of the human torso between the neck and the diaphragm or the corresponding part in other vertebrates", according to WordNet's corresponding definitional gloss). Likewise, `chest_of_drawers_n_1`, `chest_n_3`, `bureau_n_2`, and `dresser_n_2` all refer to "furniture with drawers for keeping clothes". The same holds for $S_{\texttt{chest\_n\_2}}$, even though it is a singleton set.[5]

$\textsc{Translate}(w, C, \mathrm{D}_\ell^{\ell'})$ is a function that returns a set of translations of the source word w belonging to the lexical category C using a dictionary resource, $\mathrm{D}_\ell^{\ell'}$, that contains translations from source the language, ℓ, to the target language, ℓ'.

To ease the following explanation, let us assume that that for every translation

$$\textsc{Translate}(w, C, \mathrm{D}_\ell^{\ell'}) = W' = \{w'_1, w'_2, \dots\} \tag{4.8}$$

found in the source–target dictionary there exists an inverse translation in the target–source dictionary so that for some word—or phrase—in the set of translations, $w' \in W'$, the original source word w is found in the set that represents the inverse translation of w'. This assumption can also be expressed as

$$w \in \bigcup_{w' \in W'} \textsc{Translate}(w', C, \mathrm{D}_{\ell'}^{\ell}), \tag{4.9}$$

5 A singleton set is a set with exactly one element.

where W' = Translate(w, C, $\mathrm{D}_{\ell}^{\ell'}$). We may refer to this as the assumed symmetric property of translation. It is important to note that the assumption is not universal, but tied to the structure and contents of the particular bilingual dictionary resources available; in this case the *Norsk–engelsk stor ordbok*.

Mapping Norwegian to WordNet

An example of the assumed symmetric property of translation is the case of «*brystkasse*» and "chest" in NorEng and EngNor, respectively, because

$$\begin{aligned}\textsc{Translate}(\text{«}\textit{brystkasse}\text{»}, n, \textsc{NorEng}) = \{&\text{"chest"},\\ &\text{"rib cage"},\\ &\text{"thorax"}\}\end{aligned} \tag{4.10}$$

and

$$\begin{aligned}\textsc{Translate}(\text{"chest"}, n, \textsc{EngNor}) = \{&\text{«}\textit{kiste}\text{»}, \text{«}\textit{kasse}\text{»},\\ &\text{«}\textit{skrin}\text{»}, \text{«}\textit{boks}\text{»},\\ &\text{«}\textit{bryst}\text{»}, \text{«}\textit{brystkasse}\text{»},\\ &\text{«}\textit{bringe}\text{»}, \ldots\}.\end{aligned} \tag{4.11}$$

We also see that EngNor contains other translations of "chest" too, such as «*kiste*», «*boks*», and «*skrin*», which refer to different semantic concepts than «*brystkasse*».

Some words have no direct translation into the target language. For example, Norwegian has two words for "grandmother", «*farmor*» and «*mormor*», that refer to the father's mother and to the mother's mother, respectively. Although neither «*farmor*» nor «*mormor*» can be directly translated into English, *Norsk–engelsk stor ordbok* solves this by providing "grandmother" as a translation of both «*farmor*» and «*mormor*» in NorEng and both «*farmor*» and «*mormor*» as translations of "grandmother" in EngNor. Thus, in such cases the assumption from (4.9) is still valid.

Unfortunately, there are words for which the assumed symmetric property of translation does not hold. For example, the English beverage "nog" has no direct translation into Norwegian, and none of the paraphrased translations provided by NorEng—«*eggelikør*» ("egg liqueur"), «*eggepunsj*» ("egg punch"), and «*sterkt øl*» ("strong beer")—yield "nog" when translated back to English. Nonetheless, for the rest of this discussion we choose to focus on the cases where (4.9) is valid.

Consider a translation from x to y of a given lexical category C, and that y has a nonempty set of senses

$$\text{senses}(y, C) = \langle \texttt{y\_C\_1}, \texttt{y\_C\_2}, \ldots, \texttt{y\_C\_i}, \ldots \rangle, \tag{4.12}$$

each belonging to a separate synset. Now, assume that y's ith sense, y_C_i, is a member of the synset

$$S = \{\mathtt{z}_1\mathtt{\_C\_i}_{\mathtt{z}_1}, \mathtt{z}_2\mathtt{\_C\_i}_{\mathtt{z}_2}, \dots, \mathtt{y\_C\_i}, \dots\} \tag{4.13}$$

Exploiting the Synonymy within Synsets

where each $z \in \{\mathtt{z}_j | j = 1, 2, \dots\}$ represents a word that is different from y. Hence, in the synset S those words are represented by the senses $\{\mathtt{z}_j\mathtt{\_C\_i}_{\mathtt{z}_j} | j = 1, 2, \dots\}$, which by definition are synonyms of y_C_i.

If one is able to perform a translation of the word that y_C_i represents back into the source language (an inverse translation) as described by (4.9), then that operation yields no new knowledge about whether the sense y_C_i is warranted by the semantics of x in the source language, because no new semantic knowledge has been used or discovered. Similarly, by studying the examples presented in (4.7), (4.10), and (4.11), one can see that no new knowledge about which senses of "chest" are warranted by «*brystkasse*» is gained by inverse translating "chest" from neither $S_{\mathtt{chest\_n\_1}}$, $S_{\mathtt{chest\_n\_2}}$, nor $S_{\mathtt{chest\_n\_3}}$.

However, *by definition* all the senses in synset S are semantically equivalent in some context (see sections 2.2.1 and 4.1.2). Now, let S_y represent a subset of S that only contains y_C_i. Then S'_y, the complement set of S_y, will be

$$S'_y = S - S_y = \{\mathtt{z}_j\mathtt{\_C\_i}_{\mathtt{z}_j} | j = 1, 2, \dots\}. \tag{4.14}$$

This means that the inverse translations of the words represented by the members of S'_y include x if and only if y_C_i is a member of a synset that represents a semantic concept warranted by x *and* that the following assumptions hold.

Assumption 1. *First of all, for the principle just described to hold, there must exist an inverse translation for (the word represented by) at least one of the complementary senses of* y_C_i *in* S*. That is, there must exist translations* from *what was originally the target language* back to *what was originally the source language for at least one of the senses in* S'_y*.*

Assumption 2. *Secondly, the principle requires that the synset* S*, containing the sense in question,* y_C_1*, is not a singleton set; it should actually contain other senses.*

On the other hand, if the different senses of a word y *of lexical category* C *belong to a mix of both singleton synsets and synsets with nonempty complement sets, then naturally the inverse-translation principle may be used on the nonsingleton sets. For example, given the following situation*

$$\begin{aligned} \mathit{synsets}(y, C) = \langle &\{\mathtt{y\_C\_1}\}, \\ &\{\mathtt{y\_C\_2}, \mathtt{x}_1\mathtt{\_C\_i}_{\mathtt{x}_1}, \mathtt{x}_2\mathtt{\_C\_i}_{\mathtt{x}_2}, \dots\}\rangle \end{aligned} \tag{4.15}$$

the inverse-translation principle may be used on the second synset, but not on the first one.

Assumption 3. *Thirdly, the presented principle requires that complementary senses in two or more of the synsets that each contain a different sense of a given word do not all represent a different given word.*

To elaborate on this requirement, assume that a word y, of lexical category C, has m different senses,

$$\mathit{senses}(y, C) = \langle \texttt{y\_C\_1}, \texttt{y\_C\_2}, \ldots, \texttt{y\_C\_m} \rangle \,, \tag{4.16}$$

where each sense `y_C_i` *belongs to synset S_i in*

$$\mathit{synsets}(y, C) = \langle S_1, S_2, \ldots, S_m \rangle \,, \tag{4.17}$$

then we can say that those synsets "share the word w". Now, the principle presented herein makes use of the fact that the different synsets represent different concepts. *Hence, if one of the synsets, say S_i, contains a complementary sense that represents a word that is also represented by a complementary sense in one of the other synsets, $S_j \in \mathit{synsets}(w, C)$ where $j \neq i$, there is an apparent ambiguity present. For short we may call this phenomenon shared-synonym ambiguity. It should further be noted that two shared-synonym ambiguous synsets must necessarily share* at least *two words (represented above by* `y_C_i` *and* `y_C_j`*).*

Continuing the examples from (4.7), (4.10), and (4.11), $S'_{\texttt{chest\_n\_1}}$, the complement set of $S_{\texttt{chest\_n\_1}}$ will, for example, be

$$\begin{aligned} S'_{\texttt{chest\_n\_1}} &= S_{\texttt{chest\_n\_1}} - \{\texttt{chest\_n\_1}\} \\ &= \{\texttt{thorax\_n\_2}, \texttt{pectus\_n\_1}\}, \end{aligned} \tag{4.18}$$

and by inverse translating "thorax" (from `thorax_n_2`),

$$\begin{aligned} \textsc{Translate}(\text{“thorax”}, n, \textsc{EngNor}) = \{&\text{«}\textit{bryst}\text{»}, \text{«}\textit{brystkasse}\text{»}, \\ &\text{«}\textit{toraks}\text{»}, \text{«}\textit{kropp}\text{»}, \\ &\text{«}\textit{forkropp}\text{»}\}. \end{aligned} \tag{4.19}$$

we see that the inverse translation indeed contains the original source word, «*brystkasse*».

However, by analyzing all the words and concepts in WordNet I detected that the above assumptions are not always met. The results of that analysis are presented in Section 4.7.

Algorithm 4.1: A simplified presentation of the basic mapping algorithm presented herein. Generate candidate mappings from a given word w to senses in WordNet while filtering out mappings to unwarranted senses.

```
1:  procedure MAP-WORD-TO-SENSE(w, C, D_ℓ^ℓ', D_ℓ'^ℓ)
2:      mappings ← ∅
3:      for all w' ∈ TRANSLATE(w, C, D_ℓ^ℓ') do      ▷ Translate from ℓ to ℓ'.
4:          T ← SENSES-OF(w', C)
5:          if |T| = 1 then                          ▷ Only one sense?
6:              mappings ← mappings ∪ {(w, T[0])}
7:              continue                             ▷ Skip to next translation.
8:          for all t_i ∈ T do
9:              mappings ← mappings ∪ MIRROR(w, t_i, C, D_ℓ'^ℓ)
10:     return mappings
```

Combining the Basic Principles

4.6 Combining the Basic Principles

The general principles of the method were presented above. Those principles can be combined into an algorithm for automatic mapping of words in a language different from English into WordNet senses. Algorithm 4.1 constitutes a very simplified version of the final mapping framework that will be presented in Section 4.8. However, even though the algorithm is presented in a simplified form, it serves to convey how the above principles can be combined into a coherent procedure.

The first argument to Algorithm 4.1 is the word, w, in the source language, ℓ. The second argument is the word's lexical category, C. The last two arguments to the algorithm are data structures representing the dictionary resources. The dictionary resource objects are of the same kind as the program modules NorEng and EngNor presented in Section 4.1.1, where $\mathrm{D}_{\ell}^{\ell'}$ translates from ℓ to ℓ', while $\mathrm{D}_{\ell'}^{\ell}$ translates the other way around.

The purpose of the algorithm is to return a set of correct mappings from a given word w in the source language ℓ, belonging to lexical category C, to individual senses in WordNet that are warranted by the semantics of w.

First, the set of accepted mappings is defined to be empty. Next, we call the function Translate with the arguments w, C, and $\mathrm{D}_{\ell}^{\ell'}$,

ALGORITHM 4.2: A simplified representation of the inverse translation algorithm, MIRROR, referred to by Algorithm 4.1.

```
1:  procedure MIRROR(w_O, t_i, C, D^ℓ_ℓ')
2:      mappings ← ∅
3:      S ← GET-SYNSET(t_i)
4:      S̄ ← S − {t_i}                          ▷ The complement synset.
5:      for all s ∈ S̄ do                       ▷ Complement senses.
6:          w' ← WORD-FROM-SENSE(s)
7:          for all w ∈ TRANSLATE(w', C, D^ℓ_ℓ') do
8:              if w = w_O then
9:                  mappings ← mappings ∪ {(w_O, t_i)}
10:     return mappings
```

Mapping Norwegian to WordNet

which returns all the translations of w from ℓ to ℓ'. Then, for each translation w', we let T represent the set of all senses of w' defined in WordNet, as returned by the SENSES-OF function. As indicated by the call to SENSES-OF, the set of a word's senses is constrained by the word's lexical category, C.

If T only contains one sense, we add that (see Section 4.4) to the set of accepted mappings, and immediately check the next translation. If, however, T is empty or contains multiple senses, we consider each available sense in turn. This is done by calling the function MIRROR that returns a possibly empty set of warranted mappings that are added to the set of accepted mappings. Finally, when all possible mappings for each translation have been considered, the set of accepted mappings is returned.

In the MIRROR function, shown in Algorithm 4.2, we also start with an empty set of warranted mappings, *mappings*. Then we retrieve the synset S that sense t_i belongs to, by calling the GET-SYNSET function. Next, we define a set $\overline{S}$ containing the complement senses of t_i.

Now, for each (complementary) sense s in $\overline{S}$, by calling WORD-FROM-SENSE we determine the word, w', that the sense represents and try to translate it back into ℓ. Each inverse translation, w, is then checked to see if it equals the original source word, w_O. If they are equal, the pair (w_O, t_i) is added to the set of warranted mappings. When all the complementary senses have been evaluated, the set of warranted mappings is returned.

4.7 Evaluation of the Assumptions

Evaluation of the Assumptions

To determine to what extent the assumptions mentioned in Section 4.5 are met, I wrote several scripts that traverse the complete WordNet while analyzing different aspects related to the assumptions. The results of these investigations are shown in Table 4.6 on the facing page and Table 4.7 on page 92.

For example, as can be seen from Table 4.6, 52.76 % of the noun synsets are singleton synonym sets (singleton synsets). With respect to Assumption 2, the number of singleton synsets might seem discomforting, but the table also shows that 63.65 % of those singleton synsets represent words with a single sense. Therefore, according to the principle presented in Section 4.4, the singleton-synset problem only affects 33.58 % of the noun synsets. Likewise, for the other lexical categories the singleton-synset only affects 33.42 % of the adjective synsets, 13.62 % of the verb synsets, and 43.96 % of the adverb synsets. However, the number of synsets that pose a problem according to Assumption 2 is still too large to be ignored. Countermeasures will be introduced in Section 4.8.

Furthermore, Table 4.7 shows the number of words in WordNet that are affected by the shared-synonym ambiguity described in Assumption 3. As can be seen, the fraction of words affected in the noun, adjective, and adverb categories is encouragingly small. However, the number of words affected in the verb category serves as a warning that the method might produce less precise results for verbs.

Even though none of the assumptions listed at the end of Section 4.5 are satisfied for all words and concepts in *Norsk–engelsk stor ordbok* and WordNet, the principle of inverse translating synonyms of the target sense may still be of great value. However, to handle such cases, I propose that the basic method—as sketched in Algorithm 4.1—should be augmented by exploiting the defined semantic relations available in WordNet. We may refer to this augmentation as adding search strategies to the algorithm.

4.8 Search Strategies

The main principle of the method described herein is based on the fact that synonymous senses share the same semantics, and the assumption that translations in the bilingual dictionary express the same semantics in the target language as in the source language. We may refer to such translations as lexical equivalents.

TABLE 4.6: Overview of single-sense words, singleton synsets, and singleton synsets representing single-sense words in WordNet.

Lexical category	Single-sense words	Synsets in WordNet	Singleton synsets	% of synsets	Single-sense singleton synsets[a]	% of singleton synsets	% of single-sense words
Noun	88,643[b]	73,757[c]	38,917[d]	52.76	24,769[e]	63.65	27.94
Adjective	16,889	18,877	11,750	62.25	6,309	53.69	37.36
Verb	6,261	13,650	7,942	58.18	1,859	23.41	29.69
Adverb	3,850	3,644	2,343	64.30	1,602	68.37	41.61

a A single-sense singleton synset is a synset that represents a single word that has only a single sense according to WordNet.
b 12,678 single-sense nouns that represent instances were ignored.
c 7,669 synsets representing instances were ignored.
d 2,741 singleton synsets representing instances were ignored.
e 2,083 single-sense singleton synsets representing instances were ignored.

TABLE 4.7: Words per lexical category in WordNet that are affected by shared-synonym ambiguity.

Lexical category	Words in WordNet	Words affected by shared-synonym ambiguity	% of words
Noun	117,097	3,553	3.03
Adjective	22,141	975	4.40
Verb	11,488	1,699	14.79
Adverb	4,601	116	2.52

However, as Hartmann and James (2002) comment on developing bilingual dictionaries, "finding suitable lexical equivalents is a notoriously difficult task, especially in pairs of languages with different cultures." Hence, even dictionary authors find it difficult to find the exact translations of some words. Some words and phrases are so tightly connected to the culture of the native speakers of a language that precisely translating them to other languages becomes very difficult.

One way dictionary authors handle such difficulties is by providing multiple, less accurate, translations that together convey the intended meaning. The level of accuracy for such translations may vary along several axes. One such axis designates the level of generality. This is the insight that leads to the idea to augment Algorithm 4.2 with a search algorithm that will be used when it cannot find appropriate mappings by applying only the main principles.

For example, according to NORENG the Norwegian noun *«kosthold»* translates as

$$\textsc{Translate}(\text{«}\textit{kosthold}\text{»}, n, \textsc{NorEng}) = \{\text{“diet”}, \text{“fare”}\}, \tag{4.20}$$

while WordNet define the senses of "diet" as

$$\begin{aligned} \text{synsets}(\textit{diet}, n) &= \langle S_{\texttt{diet\_n\_1}}, S_{\texttt{diet\_n\_2}}, S_{\texttt{diet\_n\_3}}, S_{\texttt{diet\_n\_4}}, \rangle \\ &= \langle \{\texttt{diet\_n\_1}, \}, \\ &\quad \{\texttt{diet\_n\_2}\}, \\ &\quad \{\texttt{diet\_n\_3}\}, \\ &\quad \{\texttt{diet\_n\_4}, \texttt{dieting\_n\_1}\}\rangle. \end{aligned} \tag{4.21}$$

We see that the first three senses of "diet" consist of singleton synsets, which pose a problem because, as noted above, singleton synsets do

Algorithm 4.3: An extended version of Algorithm 4.2.

```
1:  procedure Mirror(w_O, t_i, C, D^ℓ_ℓ')
2:      mappings ← ∅
3:      S ← Get-Synset(t_i)
4:      for all Strategy ∈ ⟨Synonym, Hypernym⟩ do
5:          S̄ ← Strategy(S, t_i)              ▷ The "complement" synset.
6:          for all s ∈ S̄ do                  ▷ Complement senses.
7:              w' ← Word-from-Sense(s)
8:              for all w ∈ Translate(w', C, D^ℓ_ℓ') do
9:                  if w = w_O then
10:                     mappings ← mappings ∪ {(w_O, t_i)}
11:         if mappings ≠ ∅ then
12:             break
13:     return mappings
```

Mapping Norwegian to WordNet

not provide enough information for the basic mirroring algorithm, Algorithm 4.2, to find any valid senses. However, WordNet defines both of the senses of "diet", `diet_n_1` ("a prescribed selection of foods") and `diet_n_3` ("the usual food and drink consumed by an organism (person or animal)"), that are semantically warranted by «*kosthold*» to be hyponyms of "fare". More importantly, EngNor provides the translation

$$\textsc{Translate}(\text{“fare”}, n, \textsc{EngNor}) = \{\ldots, \text{«}\textit{kost}\text{»}, \text{«}\textit{kosthold}\text{»}, \ldots\}, \tag{4.22}$$

which contains the original source word «*kosthold*» as one of its target words. This shows that if a synset does not provide enough information for Algorithm 4.2 to find any warranted senses, then warranted senses may be found by inverse translating senses that are, for example, hypernyms of the synset. Of course, just as for the basic algorithm, this can only work if one inverse translates words that are complementary to the original source word.

Algorithm 4.3 shows the extended version of Algorithm 4.2. The main change is how the new algorithm obtains the complement synset, $\overline{S}$. In the original version $\overline{S}$ was always defined as $S - \{t_i\}$, while the new version lets $\overline{S}$ be defined according to an arbitrary function, Strategy,

Algorithm 4.4: The Synonym-strategy function referred to by Algorithm 4.3.

```
1: procedure SYNONYM(S, t_i)
2:     return S − {t_i}                          ▷ The complement synset.
```


Algorithm 4.5: The Hypernym-strategy function referred to by Algorithm 4.3.

```
1: procedure HYPERNYM(S, t_i)
2:     S′ ← ∅
3:     for all H ∈ GET-TARGETS(S, "hypernym") do
4:         S′ ← S′ ∪ H
5:     return S′ − {t_i}                         ▷ The complement synset.
```

that takes the original synset, S, and target sense, t_i, as its arguments. As shown in the extended algorithm, the search for warranted mappings is done by iterating through a sequence of such strategy functions. The sequence of strategy functions may be altered or extended. It should be noted that for each applied strategy, if the inverse translation loop, lines 6–10 in Algorithm 4.3, does not find any warranted mappings, then the next strategy function is tried. This continues until all strategy functions have been tried, or a nonempty set of mappings has been found.

4.8.1 Synonymy and Hypernymy

The first strategy function tried in Algorithm 4.3 is Synonym, which is simply the original definition of the complement set, as shown in Algorithm 4.4. However, the Hypernym-strategy function, shown in Algorithm 4.5, discloses the reason for extending Algorithm 4.2. The idea is that instead of simply returning the t_i-complement set of S, it will replace S with all the senses that belong to hypernyms of S and store them in S'. The value returned is the t_i-complement of S'.

The extension of Algorithm 4.2 increases the method's ability to handle situations where Assumptions 1 and 2 in Section 4.5 do not hold. For example, when translating the Norwegian noun «*søk*» NorEng

suggests both "quest" and "search". Looking up the synsets of these nouns in WordNet yields

$$\text{synsets}(\textit{quest}, n) = \langle \{\texttt{quest\_n\_1}, \texttt{pursuit\_n\_2}, \texttt{pursuance\_n\_1}\}, \{\texttt{quest\_n\_2}, \texttt{seeking\_n\_1}\}\rangle \tag{4.23}$$

Mapping Norwegian to WordNet

and

$$\begin{aligned}\text{synsets}(\textit{search}, n) = \langle &\{\texttt{search\_n\_1}, \texttt{hunt\_n\_6}, \texttt{hunting\_n\_2}\},\\ &\{\texttt{search\_n\_2}\},\\ &\{\texttt{search\_n\_3}, \texttt{lookup\_n\_1}\},\\ &\{\texttt{search\_n\_4}\},\\ &\{\texttt{search\_n\_5}\}\rangle.\end{aligned} \tag{4.24}$$

Furthermore, three of five senses of the noun "search" are represented by singleton synsets, and no inverse translation that suggests *«søk»* can be found by looking up the complement senses of `quest_n_1`, `quest_n_2`, `search_n_1`, or `search_n_3` in EngNor.

This means that the original Algorithm 4.2 would not find any warranted mappings, and neither would Algorithm 4.3 if it only were to try the Synonym-strategy function. However, when the extended algorithm applies the Hypernym-strategy function, the synsets containing `quest_n_1` and `quest_n_2` are replaced by their hypernym synsets, which are shown in Figure 4.4. This, in turn, enables Algorithm 4.3 to suggest mappings by inverse translating the complement senses from the *hypernym* synsets, because EngNor suggests *«søk»* as a translation of "search".

It should be noted that the strategy algorithm Hypernym considers synsets only *one level* away from the original synset in the hypernym heterarchy. Investigating several similar cases as the one showed here revealed that searching through synsets more than one level away seldom succeeded and ran the risk of introducing less precise mappings.

4.8.2 *Hyponymy*

The preceding example shows how Algorithm 4.3 is able to find mappings missed by Algorithm 4.2 by introducing search strategies and thereby getting access to senses from hypernym synsets. However, the search strategies need not be constrained to exploiting synonymy and hypernymy relations. It seems reasonable that if—as in the preceding

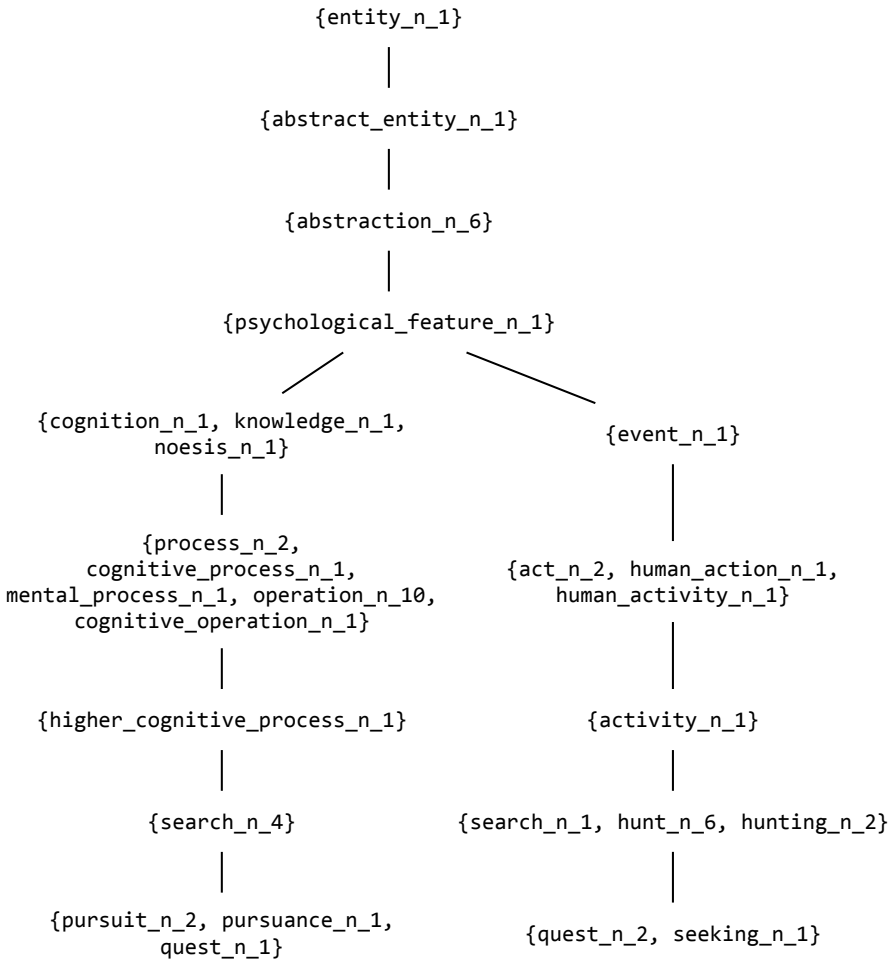


FIGURE 4.4: The hypernym-ancestor synsets of the synsets that quest_n_1 and quest_n_2 belong to.

example—otherwise missed mappings can be found by looking at hypernym synsets, adding a hyponym-strategy function might improve the results too.

For example, the basic algorithm is not able to find any mappings from the Norwegian word «*fremviser*» to senses in WordNet. NorEng suggests a single translation, namely "projector". The problem is that both the senses of "projector" are represented by singleton synsets:

Mapping Norwegian to WordNet

$$\text{synsets}(\textit{projector}, n) = \langle \{\texttt{projector\_n\_1}\}, \{\texttt{projector\_n\_2}\} \rangle. \tag{4.25}$$

Of these senses, only `projector_n_2` has hyponyms; these are shown in Figure 4.5. Now, given that Algorithm 4.3 is provided a Hyponym function—implemented just like the Hypernym function, except that it retrieves hyponyms instead of hypernyms—it will find an earlier missed mapping, namely by inverse translating `film_projector_n_1`.

4.8.3 Verb Group

The search-strategy functions presented so far are only exploiting the synonymy relationship, defined for all lexical categories in WordNet, and the hypernymy/hyponymy relationship, which is defined only for nouns and verbs. To strengthen the algorithm's ability to handle mapping of verbs, a Verb-Group search function is introduced too. As the name suggests, the new function behaves just like the Hypernym and Hyponym functions, except that it will replace the original synset by other synsets belonging to the same verb group.

For example, without the Verb-Group-strategy function, no mapping is found for the Norwegian word «*etterprøve*», which NorEng translates to "check". The reason for this is that of the synsets that represent the 25 different senses of check, 12 are singleton synsets, leaving 13, of which *not one* contain a complement sense that yields «*etterprøve*» when translated back to Norwegian. However, by applying the Verb-Group function the synset containing `check_v_3` is replaced by four semantically related synsets, where one is $\{\texttt{control\_v\_5}, \texttt{verify\_v\_2}\}$. Furthermore, the translation of "verify", using EngNor, yields «*etterprøve*» and hence shows that the use of the Verb-Group search-strategy function can improve the coverage of the algorithm.

4.8.4 Similarity

The final extension is to include a Similar-strategy function to improve Algorithm 4.3's coverage with respect to adjectives. The Similar

FIGURE 4.5: Hyponyms of projector_n_2 in the WordNet ontology.

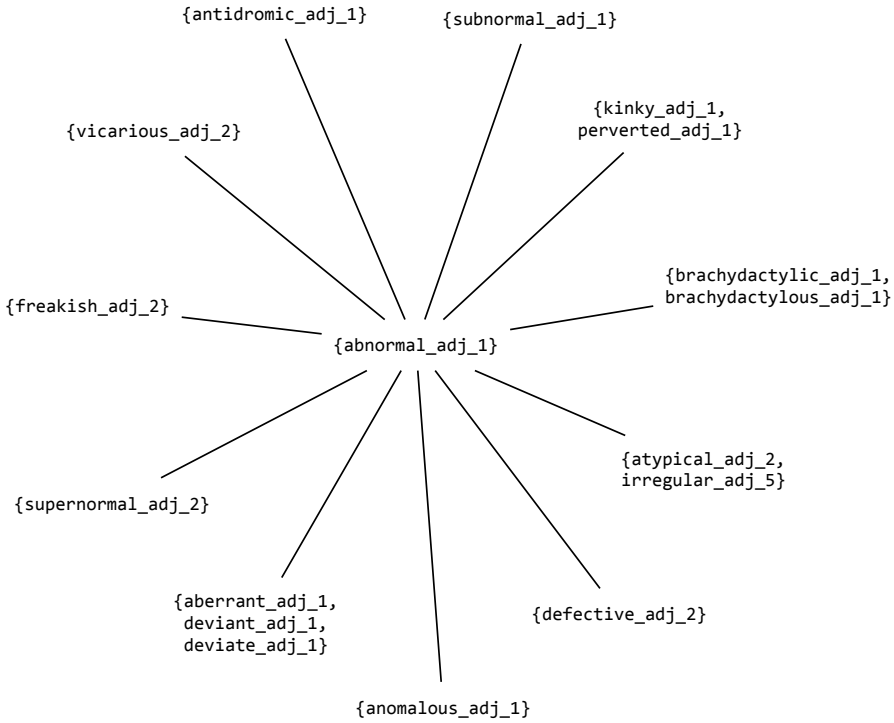


FIGURE 4.6: The synsets similar to the synset {abnormal_adj_1}.

function is implemented along the lines of the other search-strategy functions, like HYPERNYM and VERB-GROUP, but this one exploits the similar relation defined between adjective synsets. The similar relation is defined as discussed in Section 2.2.4. To see how the SIMILAR function can improve the coverage of Algorithm 4.3, please consider the case of finding an appropriate mapping for the Norwegian adjective *«abnorm»*. NORENG suggests the translation "abnormal" only. However, in WordNet the adjective "abnormal" has three different senses, all represented by singleton synsets,

Mapping Framework

$$\begin{aligned}\text{synsets}(\textit{abnormal}, \textit{adj}) = \langle &\{\texttt{abnormal\_adj\_1}\},\\ &\{\texttt{abnormal\_adj\_2}\},\\ &\{\texttt{abnormal\_adj\_3}\}\rangle.\end{aligned} \tag{4.26}$$

Since singleton synsets represent all the senses, the basic algorithm lacks any preference information. However, the application of the SIMILAR function replaces the synset containing the first sense with the semantically similar synsets shown in Figure 4.6. Of these senses, both "aberrant", "freakish", and "perverted" translate to the Norwegian *«abnorm»*. Hence, once again, the extension of Algorithm 4.2 to use search-strategy functions increases the algorithm's coverage.

4.9 MAPPING FRAMEWORK

As mentioned in Section 4.2, the method described herein generates a mapping from Norwegian words, phrases, and collocations to n-tuples of English senses as defined by WordNet, where $n \geq 1$. Below, such mappings will be referred to simply as a mapping from words in the source language to WordNet senses.

Next, the building blocks of the mapping framework will be presented, followed by examples showing how the building blocks are used to create mappings from words in the source language to WordNet senses. Appendix A provides a brief introduction to the graph theory terms used below.

The central data structure of the mapping framework is a weighted directed acyclic graph (weighted DAG) that gradually grows from a "seed" vertex—or node—representing a word in the source language, to a tree-like structure that includes all possible translations and, through them, all possible WordNet senses.

Each translational module can be seen as a function that accepts a node that represents a word in some source language, l, and expands

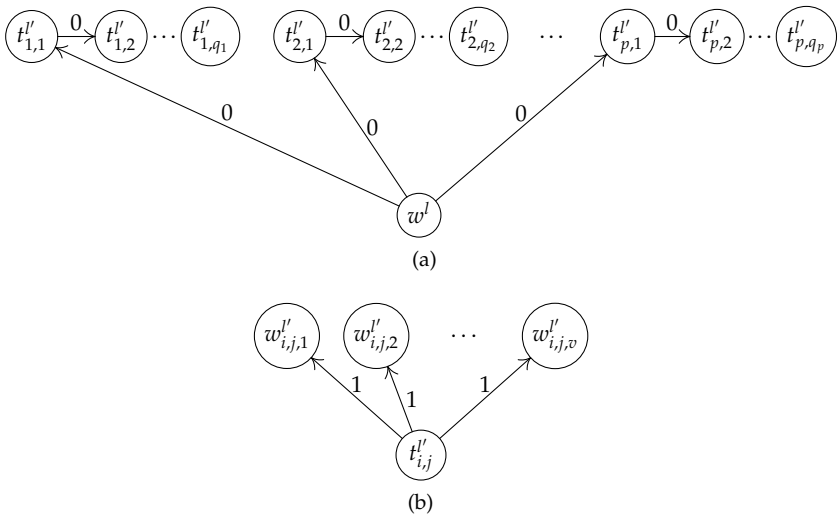


FIGURE 4.7: Expansion of translation DAG vertex types. **(a)** A vertex representing a particular word w^l, from the source language l, expands into p different translations, where each translation $t^{l'}$, in the target language l', consists of one or more parts, represented as a list of vertices. **(b)** Each such vertex, $t^{l'}_{i,j}$, representing a part of a candidate translation into the language l', expands into all v possible variants of the word $w^{l'}_{i,j}$ that $t^{l'}_{i,j}$ represents.

it by adding new edges (and nodes) to it, each representing a possible translation from l to the target language l'.

A language in this context is defined by the dictionary that is used by the translation module. For example, the modules used by the mapping framework are NORKOMPLEKS, NORENG, WordNet, and ENGNOR, in the order they are applied in the mapping process. Therefore, the first translation-step of the process is to translate a Norwegian string to the NorKompLeks "language".

Figures 4.7a and 4.7b show how a node is expanded in the general case. The input node, w^l—at the bottom of Figure 4.7a—expands into p translations, $t^{l'}$, where each translation consists of q parts. All of those nodes, called joint nodes, are connected through zero-weighted edges. In turn, each part, $t^{l'}_{i,j}$ where $i = 1, 2, \ldots, p$ and $j = 1, 2, \ldots, q_p$, of each

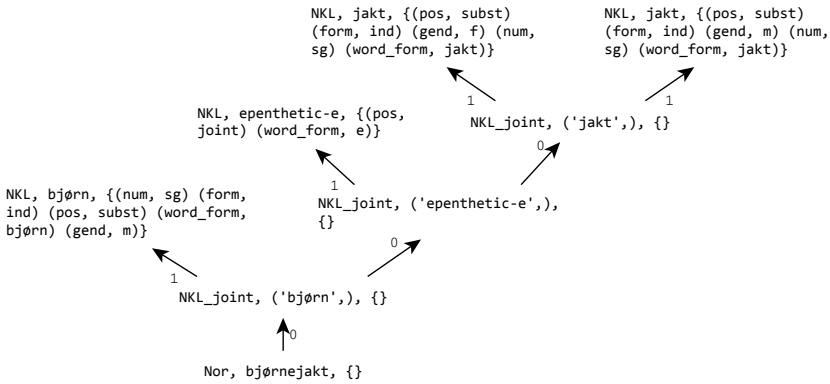


FIGURE 4.8: Example of how a weighted DAG is expanded in the case of a multipart translation.

translation is expanded in v alternatives, as shown in Figure 4.7b. These alternatives are connected through edges with a weight of 1.

The general case of node expansion is designed to handle multipart translations in order to, for example, handle translations of compound words that are not found in the machine-readable dictionary used by the translation module in question. For example, Figure 4.8 shows how the translation of the compound «*bjørnejakt*» ("bear hunt") is handled by the translation module NORKOMPLEKS.[6]

Each edge from the weighted DAG's source represents a possible translation of the given input. We may view each such edge as a *branch* of a tree. However, since the branch represents a subgraph of a directed acyclic graph (DAG), it may join with other branches at a vertex further down its own path; this is the reason the final structure is called *tree-like*.

The reason for using the weighted edges and joint nodes is to be able to maintain the ordering between the constituents when a translation requires that a word is split up, while keeping the algorithms for traversing and inspecting the DAG simple. For example, the (weighted) distance from the source node to all the NorKompLeks translated nodes in Figure 4.8 is 1.

6 As mentioned in Chapter 3, the NorKompLeks module features a compound-word analyzer that will try to split up words not found in the dictionary, such as «*bjørnejakt*».

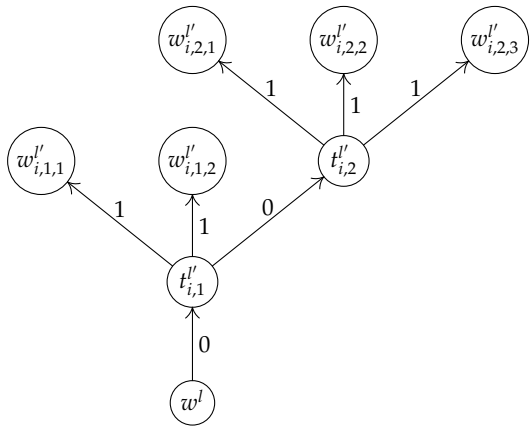


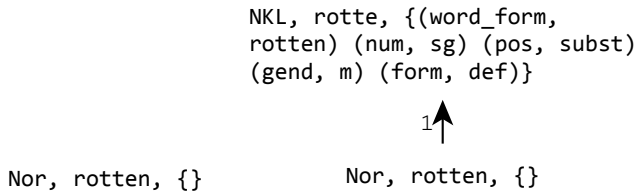
FIGURE 4.9: Example of a generalized subgraph representing a multipart translation.

Furthermore, the target nodes of each joint node represent alternative translations of that part. Each such set of alternatives can be represented as a set of nodes. Hence, to construct all the different translations of a subgraph like the one in Figure 4.8, one only needs to compute the Cartesian product of the sets representing the parts. For example, the different translations represented by the graph in Figure 4.9, can be represented by $\{w^{l'}_{i,1,1}, w^{l'}_{i,1,2}\} \times \{w^{l'}_{i,2,1}, w^{l'}_{i,2,2}, w^{l'}_{i,2,3}\}$, which evaluates to

$$\{(w^{l'}_{i,1,1}, w^{l'}_{i,2,1}), (w^{l'}_{i,1,1}, w^{l'}_{i,2,2}), (w^{l'}_{i,1,1}, w^{l'}_{i,2,3}),$$
$$(w^{l'}_{i,1,2}, w^{l'}_{i,2,1}), (w^{l'}_{i,1,2}, w^{l'}_{i,2,2}), (w^{l'}_{i,1,2}, w^{l'}_{i,2,3})\},$$

where each tuple represents one possible concatenation of the variants of the parts of a multipart translation.

However, because most translations consist of single words, the use of multipart expansions only would result in graphs with a lot of superfluous nodes; that is, joint nodes with only one target—or child—node. Therefore, the translational modules also expand nodes in a more compact way, by simply leaving out the zero-weighted edge and joint node if the joint node has only one target node. In the following example, several such branches will be used.



(a) Initial seed.

(b) Second step.

FIGURE 4.10: The two first steps of the mapping of *«rotten»*.

4.9.1 Example Mapping of «rotten»

This section will present a complete example of a mapping from the Norwegian word *«rotten»* ("rat.the"). The definite form is used to distinguish the input word from its ambiguous stem *«rotte»*, which can be used to refer to both the noun and a verb; however, the verb means to "conspire against" or "gang up on" and therefore differs from the English verb "rat".[7]

Given ambiguous input—like *«rotte»*—the framework will handle both the nominal and the verbal meaning in separate subgraphs, just like any other ambiguous translation. The distinction in this case was made only to make the example more comprehensible.

Creating the Graph

Figure 4.10a shows the input to the mapping framework, also known as the seed node. The seed node represents a tuple containing a language identifier, the input word itself, and a set of attributes that describe the input word. The seed node's attribute set is always empty.

The second step of the mapping process is shown in Figure 4.10b. The figure shows how the DAG is expanded to represent the "translation" to NorKompLeks. As shown in the figure, the NorKompLeks nodes contain a nonempty set of attributes. Furthermore, the second element

7 The verb "rat" has several meanings. For example, WordNet uses the glosses "desert one's party or group of friends, for example, for one's personal advantage", "take the place of work of someone on strike", and "give away information about somebody" to describe some of them.

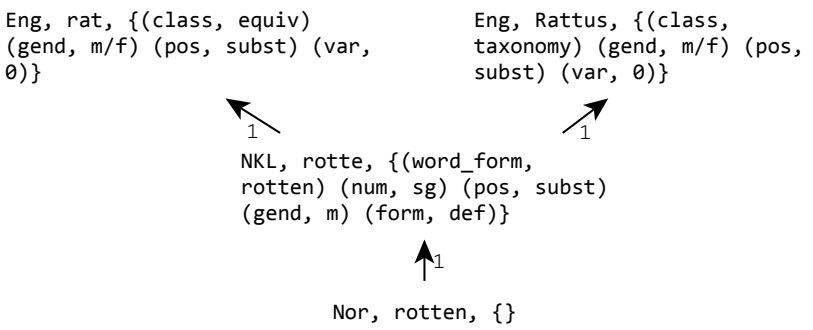


FIGURE 4.11: The third step of the mapping of *«rotten»*.

of the tuple represents the stem of the source word, while the value of the *word form* attribute represents the word form of the source word.

The translation from Norwegian—with characteristics retrieved from NorKompLeks—to English is shown in Figure 4.11. The translation is performed by the NORENG module by looking up the stem of the source node. The module uses information about the source word's lexical class, designated by the source node's *pos* attribute, to restrict the set of possible translations. For example, NORENG lists both verbal and a nominal translations of the stem *«rotte»*. However, because the source node represents a noun, only the nominal translations will be used in the expansion.

Next, each of the English nodes are "translated" into matching WordNet senses, as shown in Figure 4.12. The language designator and attribute set has been removed in the figure, both to save space and because the lexical class and sense number are codified into the string representing the node.

The next step, shown in Figure 4.13, is to expand the WordNet sense nodes by adding nodes that represent the WordNet synsets that each sense belongs to.

Sense Evaluation

After creating the initial translation graph, the mapping framework must decide which senses are warranted by the original input word. The

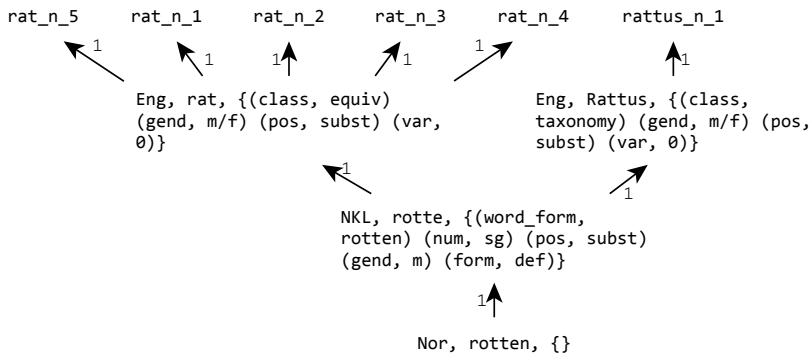


FIGURE 4.12: The DAG after adding the WordNet sense nodes.

decisions are made by applying functions that implement the principles developed in Sections 4.2–4.8.

By evaluating the graph and applying the principle that mappings that represent *single-sense* words are kept without further ado (Section 4.4) tells the mapping module that the translation of «*rotten*» into the sense `rattus_n_1` is warranted. Hence, the framework returns a tuple including that sense:

$$
\begin{array}{l}
(\texttt{rotten},\\
\ \langle((\texttt{rat\_n\_1},\\
\quad(\{(\texttt{form},\texttt{def}),(\texttt{gend},\texttt{m}),(\texttt{num},\texttt{sg}),(\texttt{pos},\texttt{subst}),\\
\qquad(\texttt{status},\texttt{nf}),(\texttt{word\_form},\texttt{rotten})\},\texttt{rotte},48957)),),\\
\quad\ldots,\\
\rangle,).
\end{array}
\tag{4.27}
$$

If none of the search strategies mentioned in Section 4.8 had been applied, then that sense would have been the only one returned. The reason for this is that the other WordNet senses in the graph all represent the noun "rat", and the senses `rat_n_1` and `rat_n_5` are both represented by singleton synsets (see Assumption 2 in Section 4.5), while applying the MIRROR algorithm *without* any search strategies—as described in Algorithm 4.2—on the remaining senses fails to find any warranted mappings.

FIGURE 4.13: The DAG after adding the WordNet synset nodes.

TABLE 4.8: Synsets and descriptions for the five WordNet senses of "rat".

Sense #	Synset	Description
1	rat	any of various long-tailed rodents similar to but larger than a mouse
2	scab, strikebreaker, blackleg, rat	someone who works (or provides workers) during a strike
3	rotter, dirty dog, rat, skunk, stinker, stinkpot, bum, puke, crumb, lowlife, scum bag, so-and-so, git	a person who is deemed to be despicable or contemptible; "only a rotter would do that"; "kill the rat"; "throw the bum out"; "you cowardly little pukes!"; "the British call a contemptible person a 'git"'
4	informer, betrayer, rat, squealer, blabber	one who reveals confidential information in return for money
5	rat	a pad (usually made of hair) worn as part of a woman's coiffure

As shown by Table 4.8, which describes the synsets that the different senses of "rat" belong to, not finding any other warranted mappings is the desired behavior for the framework, *except* for the first sense.

However, if the HYPONYM search strategy—described in Section 4.8.2—is used with the extended MIRROR function—described in Algorithm 4.3—the mapping framework will extend the graph by expanding the WordNet synset nodes of the graph, so that their hyponym synsets are included. Figure 4.14 shows how the {`rat_n_1`} node is expanded. Furthermore, NORENG translates both "brown rat" and "Norway rat" to «*rotte*». Therefore, the mapping to `rat_n_1` is also warranted, and the

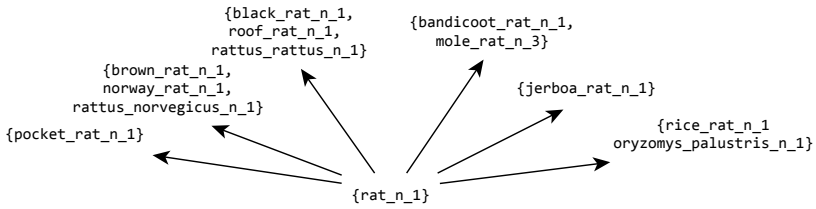


FIGURE 4.14: Expansion of the graph by adding nodes representing hyponyms of the {rat_n_1} node.

complete returned value from the framework, given «*rotten*» as input, becomes

$$
\begin{aligned}
&(\texttt{rotten},\\
&\langle((\texttt{rat\_n\_1},\\
&\quad(\{(\texttt{form},\texttt{def}),(\texttt{gend},\texttt{m}),(\texttt{num},\texttt{sg}),(\texttt{pos},\texttt{subst}),\\
&\qquad(\texttt{status},\texttt{nf}),(\texttt{word\_form},\texttt{rotten})\},\texttt{rotte},48957)),),\\
&\quad((\texttt{rattus\_n\_1},\\
&\quad(\{(\texttt{form},\texttt{def}),(\texttt{gend},\texttt{m}),(\texttt{num},\texttt{sg}),(\texttt{pos},\texttt{subst}),\\
&\qquad(\texttt{status},\texttt{nf}),(\texttt{word\_form},\texttt{rotten})\},\texttt{rotte},48957)),)\rangle,).
\end{aligned}
\tag{4.28}
$$

This example has shown how the framework implementing the principles described in Sections 4.2–4.8 is able to find warranted mappings from a Norwegian noun to particular concepts in WordNet. The example also showed how the use of a search strategy—namely, HYPONYM—was crucial to find one of the warranted mappings.

4.10 RESULTS

To test Verto, the framework for finding warranted mappings of Norwegian words to senses in WordNet, several experiments were run. The experiments were designed to shed light on several aspects of the method in order to answer the research questions (see Section 1.3). The aspects include:

1 In total, and for each lexical category, how many of the keywords in *Norsk–engelsk stor ordbok* is the method able to find mappings for?

2 How many mappings does the method generate (considering that a single keyword can result in multiple mappings)?

3 How precise is the method?

4 What are the reasons for missing mappings?

Results

5 How many of the total of warranted mappings does the method find?

6 How does changing the parameters—that is, the search strategies—of the method affect the above factors?

The set of mappings from Norwegian words to WordNet senses produced by running Verto constitutes a novel semantic resource, for reasons of simplicity named Ordnett[8].

4.10.1 Measures

To measure the precision of the mappings suggested by Verto, the well-known precision measure, from the fields of Information Retrieval (IR) and Information Extraction (IE) (Jurafsky and Martin 2000; p. 578), was adapted. If we let P represent the precision, we have

$$P = \frac{|A^+ \cap S|}{|S|}, \tag{4.29}$$

where A^+ represents the set of all the correct answers and S is the set of answers given by the system.

Each mapping suggested by Verto constitutes an answer. In the case that a Norwegian compound is split up and each constituent is internally mapped individually, or if a Norwegian simplex word translates to an English multiword expression, Verto suggests a mapping from the input word to a *tuple* of senses. Each such mapping counts as one answer. For such a mapping to be counted as correct, all the elements of the tuple must be correct. Thus, the precision measure describes how many of the mappings returned by the system are actually correct.

It should be noted that the denominator above includes both correct and incorrect answers. Hence, if all the answers are correct, P will be 1. In the following evaluations, a correct answer is considered a correct suggested mapping.

8 Ordnett means word net in Norwegian. However, the new semantic resource should not be confused with Kunnskapsforlaget's Web site, `http://www.ordnett.no/`, which provides access to on-line dictionaries.

Another well known measure from the fields of IR and IE is the recall measure, defined as

$$R = \frac{|A^+ \cap S|}{|A^+|}, \tag{4.30}$$

which describes how many of all the possible correct answers were returned from the program.

When measuring a system with the precision and recall measures, modifying the system to increase its precision score often leads to a decrease in its recall score. This phenomenon has lead researchers to introduce a measure that incorporates the precision and recall scores in a balanced way. The measure, named F-measure, is defined as

$$F = \frac{(\beta^2 + 1)PR}{\beta^2 P + R}, \tag{4.31}$$

where β is a weight to balance the importance between precision, P, and recall, R. If β equals 1, precision and recall are weighted equally important. When $\beta > 1$, recall is favored, and $\beta < 1$ favors precision[9].

Some commonly used β values give rise to named F-measures, like $F_{0.5}$, where $\beta = 0.5$, F_1, where $\beta = 1$, F_2, where $\beta = 2$, etc.

It should be noted that the system being tested cannot itself decide what constitutes a correct answer; if it could, all answers would be correct. One way to decide which answers are correct is to use a human expert to generate a test set that also defines the total number of possible correct mappings. This approach was used in the evaluation of Verto, and is described in the next section.

4.10.2 *The Test Set*

To test the quality of the suggested mappings, a test set was generated for each lexical category; that is, noun, adjective, verb, and adverb. For each of the categories, a selection of randomly chosen keywords from NorEng was made, based on a uniform probability distribution, using the Mersenne Twister pseudo-random generator (Matsumoto and Nishimura 1998).

I developed a Web application that given a Norwegian word as input would present all the target WordNet senses/synsets that are *reachable* by Verto—based on the available resources presented in the previous chapter—to a human expert. Here, reachable means *all* the WordNet

9 Contrary to what Jurafsky and Martin (2000; p. 578) state. They claim that when $\beta > 1$ precision is favored, and vice versa.

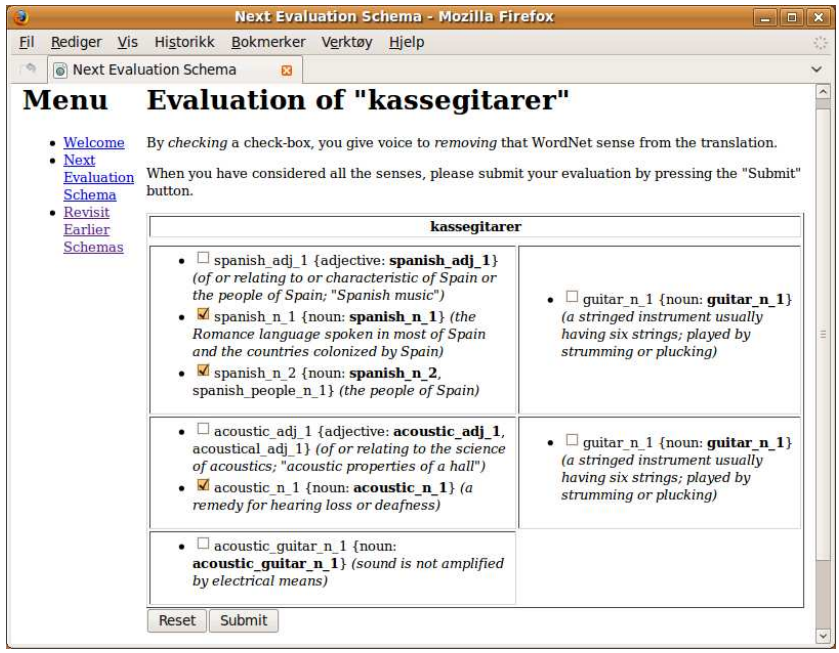


FIGURE 4.15: The Web interface used by the human expert to define the test sets. The human expert marks the check-boxes for senses that he or she believes are not warranted by the semantics of the original Norwegian word.

target synset nodes that are included in the DAG used by Verto during the mapping process. This way, the senses presented to the human expert were the same as the ones the Verto algorithm has to consider.

Figure 4.15 on the facing page shows a form where the original Norwegian word along with possible target senses and their glosses are presented to a human expert by the Web application. All the possible senses for each particular word were gathered in one cell in the table, and each word in a multipart expression contributed to one column. Thus, if all the words were simplex expressions (which they were not), the table would consist of a single column, but with multiple rows.

To generate the test set, a Norwegian Bachelor's level student finishing his Master's degree, with three and a half years of English studies at the university level, was appointed to be the human expert, based on recommendations from *Mila Dimitrova-Vulchanova*, professor at the Department of Modern Languages at the Norwegian University of Science and Technology (NTNU). Unfortunately, at the time the test set was created Dimitrova-Vulchanova could only recommend this sole student that she was convinced had a good enough understanding of *both* Norwegian and English to create a test set of adequately high quality. Therefore, opportunities to take advantage of possible inter-annotator agreement were missed. The human expert was paid 150 Norwegian kroner per hour for his effort.

The human expert had no idea of the workings of the algorithm, and was simply asked to evaluate all the senses presented to him, and whether each sense was warranted by the semantics and use of the Norwegian "seed" word that was presented along with the sense. If a sense was not warranted, the expert marked this by removing the check mark in the check-box for that sense; this was done with a single mouse click.

Furthermore, if a mapping consisted of several parts—that is, a compound mapping—then *all* its constituents had to be considered correct to consider the whole mapping correct.

The human expert was allowed to consult all the dictionaries and encyclopedias he wanted to; the overriding concern was that he was confident that the answers he gave were correct.

Table 4.9 on the next page shows the number of words and senses the human expert evaluated, as well as their distribution over the different lexical categories. A bigger test set may be desirable, but the amount of work to create the test set described herein was quite substantial. The human expert worked for 33.5 hours to create the test set.

Table 4.9: Number of words and senses considered by the human expert.

Lexical category	Words	Senses
Noun	86	1,382
Adjective	97	1,759
Verb	62	2,884
Adverb	64	729
Total	309	6,754

4.10.3 The Experiment

In the experiment, Verto was run with all the keywords in *Norsk–engelsk stor ordbok* as input, with twelve different permutations of parameters controlling the program.

It should be noted that by running the test on such a large set of input words, a product of the experiment is the creation of a resource that consists of mappings from Norwegian words to WordNet senses.

One of the parameters was the dictionary resources available to Verto. In the first six test runs the *original Norsk–engelsk stor ordbok* and *Engelsk–norsk stor ordbok* dictionaries were used in the form of the NorEng and EngNor modules, respectively. In the last six test runs two new, *expanded* modules, NorEng$_C$ and EngNor$_C$, were used instead. NorEng$_C$ was created by inverting all the translations in *Engelsk–norsk stor ordbok* and adding them to *Norsk–engelsk stor ordbok*; thus, creating a *combined* version of *Norsk–engelsk stor ordbok*. EngNor$_C$ was made in the same fashion, except that inverted *Norsk–engelsk stor ordbok* translations were added to *Engelsk–norsk stor ordbok*.

Adding inverted translations might seem like a strange thing to do under the assumption that the same translations occur both in the Norwegian–English and in the English–Norwegian dictionary. However, that assumption does not hold; for example there is an entry in NorKompLeks for the prefix «*anti-*» and an entry in *Engelsk–norsk stor ordbok* for "anti", but none of these entries are found in *Norsk–engelsk stor ordbok*. Thus, by defining the new, combined dictionary resources NorEng$_C$ and EngNor$_C$, the number of possible translations in both directions is increased.

Table 4.10: Search strategies used for each lexical category in each per-dictionary partition of the experiment. The numbers represent the different test runs of the partition.

Lexical category	Synonym	Hypernym	Hyponym	Similar	Verb-group
Noun	1–6	2, 6	3, 6		
Adjective	1–6	2, 6	3, 6	4, 6	
Verb	1–6	2, 6	3, 6		5, 6
Adverb	1–6	2, 6	3, 6		

The dictionary-resources parameter naturally divides the twelve test runs of the experiment into two partitions of six test runs each. Each partition comprises six test runs where the parameters that control the search strategies of the extended Mirror function (see Algorithm 4.3 on page 93) vary according to Table 4.10. Each number in the table indicates a test run relative to the dictionary-resource partitioning. For example, in all the test runs—that is, 1–6 and 7–12—the basic Synonym search strategy was used, while the Similar search strategy was employed in test runs 4, 6, 10, and 12. Furthermore, it should be noted that test runs number 6 and 12 employed *all* the different search strategies.

4.10.4 *With the Original Dictionaries*

This section will present the results of the six first test runs. That means that all of the test runs documented in this section were performed with the original, pristine dictionary modules NorEng and EngNor.

Test Run 1 (The Synonym Strategy)**.** The first test run represents a starting point where only the Synonym strategy, presented in Section 4.8.1, is applied.

The results of each test run are represented by two tables; one table presents values related to the method's coverage, the other table presents precision, recall, and $F_{0.5}$-measure values.

The coverage of the method describes how many of the words in the source dictionary, *Norsk–engelsk stor ordbok*, the method was able to find any mapping for. Table 4.11 on the following page shows the coverage values for the first test run.

TABLE 4.11: Coverage results of running Verto with the original dictionaries, and the SYNONYM strategy.

Lexical category	Keywords	Mapped		No inverse		Mappings	
		N	(%)[a]	*N*	(%)[b]	*N*	Mean
Noun	45,487	22,841	50.2	17,727	78.3	44,104	0.970
Adjective	9,197	3,304	35.9	4,710	79.9	8,290	0.901
Verb	5,359	3,495	65.2	1,654	88.7	13,750	2.566
Adverb	583	186	31.9	285	71.8	458	0.786
Total	60,626	29,826	49.2	24,376	79.1	66,602	1.306

a Percentage of keywords mapped to WordNet senses.
b Percentage of cases where *no* mapping could be found.

TABLE 4.12: Number of times that no mapping could be found due to missing WordNet entries, with pristine dictionaries.

Lexical category	Word not in WordNet
Noun	4,919
Adjective	1,183
Verb	210
Adverb	112
Total	6,424

The first column of the coverage table shows the number of input words in each lexical category and the total number of input words.

The second column shows the number of words for which Verto was able to find *one or more* mapping for, with the percentage in the third column.

It should be noted that there are basically two reasons for not finding any mappings for an input word. The first reason is that the graph could not be expanded to include any WordNet senses. The input words in each test run are the entries in the dictionaries NORENG and NORENG_C for the first six test runs and last six test runs, respectively. Therefore, if the graph does not contain any WordNet senses, it is because WordNet does not include any of the words needed to extend the graph.

Table 4.13: Precision, recall, and $F_{0.5}$ results of running Verto with the original dictionaries and the Synonym strategy.

	Mappings				
Lexical category	Correct	Incorrect	Precision	Recall	$F_{0.5}$
Noun	126	7	0.947	0.217	0.566
Adjective	121	3	0.976	0.186	0.527
Verb	160	26	0.860	0.161	0.460
Adverb	110	10	0.917	0.308	0.657
Total	517	46	0.918	0.200	0.535

Table 4.12 on the preceding page shows the number of words for which no mapping could be found because there was no entry in WordNet that could be included in the translation graph. Since these numbers only depend on the dictionary used, they are equal for all of the six first test runs.

The second reason for not being able to find a mapping for a word is being unable to find a *warranted* mapping; that is, if the extended Mirror algorithm is unable to return any mappings. The extended Mirror algorithm's ability to find inverse translations depends on which search-strategy functions are used.

The fourth column of the coverage table shows the number of words for which no mapping could be found because no inverse translation could be found. The percentage, in column five, represents the fraction of cases were no mapping could be found. This was caused by the extended Mirror algorithm not being able to find a justification for the mapping.

From Table 4.11 on the facing page the method seems most effective when finding mappings for verbs (65.2 %) and nouns (50.2 %). Furthermore, it seems that the average number of mappings *per input word* is much higher for verbs than for the other lexical categories.

The percentage of mapped words is 49.2 % while the mean number of mappings per word is 1.306, averaged over the lexical classes.

Table 4.13 shows the correct/incorrect classification of each found mapping, along with the precision, recall and $F_{0.5}$ values for Verto based on the results of the test run. These values are computed by comparing the output from Verto with the test set generated by the human expert.

TABLE 4.14: Coverage results of running Verto with the original dictionaries and the SYNONYM and HYPERNYM search strategies.

Lexical category	Keywords	Mapped		No inverse		Mappings	
		N	(%)[a]	*N*	(%)[b]	*N*	Mean
Noun	45,487	23,253	51.1	17,315	77.9	48,481	1.066
Adjective	9,197	3,305	35.9	4,709	79.9	8,295	0.902
Verb	5,359	3,613	67.4	1,536	88.0	17,816	3.325
Adverb	583	188	32.2	283	71.6	460	0.789
Total	60,626	30,359	50.1	23,843	78.8	75,052	1.520

a Percentage of keywords mapped to WordNet senses.
b Percentage of cases where *no* mapping could be found.

The method's recall score, or ability to find mappings for each input word, is somewhat low, but it is encouraging to see that the precision of the method is quite high; as high as 0.976 for adjectives. The lowest precision is for verbs at 0.860.

Because the output from Verto should be usable by other computer programs, a high level of precision is more important than the recall score. Therefore, the tables report an F-measure where $\beta = 0.5$.

Test Run 2 (The SYNONYM and HYPERNYM Strategies)**.** In the second test run, the SYNONYM and HYPERNYM search strategies, both presented in Section 4.8.1, comprise the extended MIRROR function's repertoire.

As can be seen from Table 4.14, there was an increase in the number of mapped nouns (+ 0.9 %), verbs (+ 2.2 %), and adverbs (+ 0.3 %), while the number of mapped adjectives stayed the same as in the first test run. Consequently, the number of cases where no inverse translation could be found decreased correspondingly.

The average number of mappings per input verb increased quite a lot.

Table 4.15 on the next page shows that the precision generally decreased and the recall generally increased, except for the adjective category, which stayed unchanged.

It should also be noted that the total number of correctly and incorrectly mapped words in Table 4.15 on the facing page did increase, due to the increased recall.

TABLE 4.15: Precision, recall, and $F_{0.5}$ results of running Verto with the original dictionaries and the SYNONYM and HYPERNYM search strategies.

	Mappings				
Lexical category	Correct	Incorrect	Precision	Recall	$F_{0.5}$
Noun	137	8	0.945	0.236	0.590
Adjective	121	3	0.976	0.186	0.527
Verb	189	32	0.855	0.190	0.503
Adverb	111	10	0.917	0.311	0.660
Total	558	53	0.913	0.216	0.555

TABLE 4.16: Coverage results of running Verto with the original dictionaries and the SYNONYM and HYPONYM search strategies.

Lexical		Mapped		No inverse		Mappings	
category	Keywords	*N*	(%)[a]	*N*	(%)[b]	*N*	Mean
Noun	45,487	23,451	51.6	17,117	77.7	49,024	1.078
Adjective	9,197	3,304	35.9	4,710	79.9	8,292	0.902
Verb	5,359	3,707	69.2	1,442	87.3	17,507	3.267
Adverb	583	186	31.9	285	71.8	458	0.786
Total	60,626	30,648	50.6	23,554	78.6	75,281	1.508

a Percentage of keywords mapped to WordNet senses.
b Percentage of cases where *no* mapping could be found.

Test Run 3 (The SYNONYM and HYPONYM Strategies)**.** In test run three the HYPERNYM strategy was replaced by the HYPONYM strategy, which was presented in Section 4.8.2.

As shown in Table 4.16, the change caused an increase in the number of mapped words for two of the lexical categories, leading to a temporarily new maximum for the noun (+ 1.5 %) and verb (+ 4.0 %) categories. The parenthetic change indicators (+/-) are relative to the first test run unless something else is explicitly stated.

Furthermore, the table shows the highest total percentage of words mapped so far (at 50.6 %).

In the previous test run, the precision score decreased for all the lexical categories where the number of mappings—and thus the recall—increased. However, as shown in Table 4.17 on the following page,

TABLE 4.17: Precision, recall, and $F_{0.5}$ results of running Verto with the original dictionaries and the SYNONYM and HYPONYM search strategies.

	Mappings				
Lexical category	Correct	Incorrect	Precision	Recall	$F_{0.5}$
Noun	147	8	0.948	0.253	0.612
Adjective	121	3	0.976	0.186	0.527
Verb	192	41	0.824	0.193	0.498
Adverb	110	10	0.917	0.308	0.657
Total	570	62	0.902	0.221	0.558

TABLE 4.18: Coverage results of running Verto with the original dictionaries and the SYNONYM and SIMILAR search strategies.

Lexical		Mapped		No inverse		Mappings	
category	Keywords	*N*	(%)[a]	*N*	(%)[b]	*N*	Mean
Noun	45,487	22,888	50.3	17,680	78.2	44,363	0.975
Adjective	9,197	4,387	47.7	3,627	75.4	13,753	1.495
Verb	5,359	3,495	65.2	1,654	88.7	13,750	2.566
Adverb	583	186	31.9	285	71.8	458	0.786
Total	60,626	30,956	51.1	23,246	78.3	72,324	1.456

a Percentage of keywords mapped to WordNet senses.
b Percentage of cases where *no* mapping could be found.

during Test Run 3 both the precision and the recall increased for the noun category.

On the other hand, a decrease in the precision of verb mappings caused a decrease in the overall precision score. Nonetheless, the total $F_{0.5}$ score scarcely reached a new maximum, due to an increased overall recall score.

Test Run 4 (The SYNONYM and SIMILAR Strategies)**.** The fourth test run applied the basic SYNONYM strategy along with the SIMILAR search strategy that was presented in Section 4.8.4.

Table 4.18 shows that this lead to a strong increase in the number of mapped adjectives (+ 11.8 %), which caused the overall number of mapped words to reach a new maximum (at 51.1 %).

TABLE 4.19: Precision, recall, and $F_{0.5}$ results of running Verto with the original dictionaries and the SYNONYM and SIMILAR search strategies.

	Mappings				
Lexical category	Correct	Incorrect	Precision	Recall	$F_{0.5}$
Noun	129	7	0.949	0.222	0.573
Adjective	219	10	0.956	0.336	0.698
Verb	160	26	0.860	0.161	0.460
Adverb	110	10	0.917	0.308	0.657
Total	618	53	0.921	0.239	0.587

As shown in Table 4.19, just as in the previous test run, both the precision and recall scores for the noun category increased compared to the first test run. However, while the precision reached yet another maximum (at 0.949), the recall did not increase as much as in Test Run 3.

The strong increase in the number of mappings of adjectives caused the corresponding recall number to increase strongly as well. Not surprisingly, the adjective precision score dropped slightly.

As a result, a new overall maximum in both precision (at 0.921) and recall (at 0.239) caused yet another maximum $F_{0.5}$ score (at 0.587).

Test Run 5 (The SYNONYM and VERB-GROUP Strategies)**.** The fifth test run replaced the SIMILAR search strategy with the VERB-GROUP strategy presented in Section 4.8.3.

According to Table 4.20 on the next page, the strategy seems to have little effect on the coverage in general. In fact, only two more words were mapped, both verbs. However, the number of mappings per verb increased quite a bit (+ 933). Except for that, the search strategy had an insignificant impact on the number of words mapped.

Table 4.21 on the following page shows no change in the precision and recall scores compared to the first test run, except for the verb category, where precision decreased (- 0.008) slightly and recall increased (+ 0.013). The net effect was a minute decrease in the $F_{0.5}$ score (- 0.005).

Test Run 6 (The SYNONYM and All Search Strategies)**.** The sixth test run represents the end of the first partition of test runs. In this test run, *all* the search strategies were included in the extended MIRROR algorithm's repertoire.

TABLE 4.20: Coverage results of running Verto with the original dictionaries and the SYNONYM and VERB-GROUP search strategies.

Lexical category	Keywords	Mapped		No inverse		Mappings	
		N	(%)[a]	*N*	(%)[b]	*N*	Mean
Noun	45,487	22,841	50.2	17,727	78.3	44,104	0.970
Adjective	9,197	3,304	35.9	4,710	79.9	8,290	0.901
Verb	5,359	3,497	65.3	1,652	88.7	14,683	2.740
Adverb	583	186	31.9	285	71.8	458	0.786
Total	60,626	29,828	49.2	24,374	79.1	67,535	1.349

a Percentage of keywords mapped to WordNet senses.
b Percentage of cases where *no* mapping could be found.

TABLE 4.21: Precision, recall, and $F_{0.5}$ results of running Verto with the original dictionaries and the SYNONYM and VERB-GROUP search strategies.

Lexical category	Mappings		Precision	Recall	$F_{0.5}$
	Correct	Incorrect			
Noun	126	7	0.947	0.217	0.566
Adjective	121	3	0.976	0.186	0.527
Verb	173	30	0.852	0.174	0.479
Adverb	110	10	0.917	0.308	0.657
Total	530	50	0.914	0.205	0.540

As Table 4.22 on page 124 shows, the results include a new maximum coverage score for both nouns (at 52.1 %) and verbs (at 70.2 %), while the corresponding scores for adjectives and adverbs are equal to the maximums found earlier in Table 4.18 on page 120 and 4.14 on page 118, respectively. Consequently, the total coverage score also reached a new maximum (at 52.9 %).

Not surprisingly, as shown in Table 4.23 on page 124, all the precision scores were lower than in the first test run, except for the adverb category, which was unchanged. As a result, the overall precision score reached a new minimum (at 0.899).

Conversely, the overall recall score reached a new maximum (at 0.277), and so did the $F_{0.5}$ score as well (at 0.621).

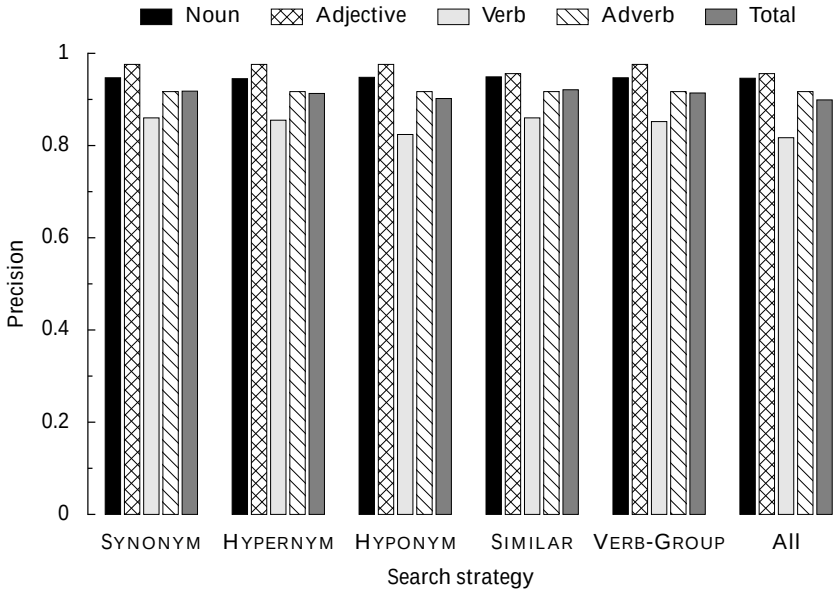


FIGURE 4.16: Summary of precision scores through test runs 1–6.

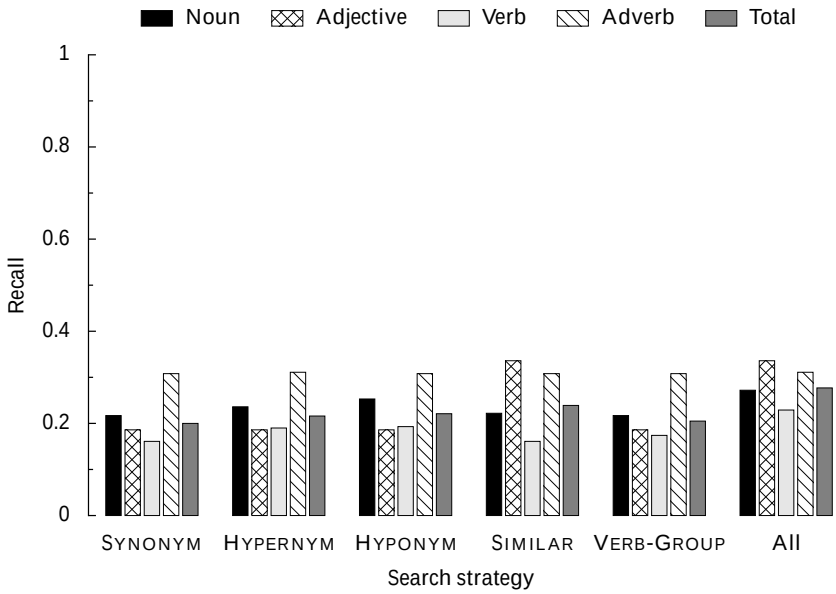


FIGURE 4.17: Summary of recall scores through test runs 1–6.

TABLE 4.22: Coverage results of running Verto with the original dictionaries and the SYNONYM and all of the search strategies.

Lexical category	Keywords	Mapped		No inverse		Mappings	
		N	(%)[a]	*N*	(%)[b]	*N*	Mean
Noun	45,487	23,720	52.1	16,848	77.4	53,344	1.173
Adjective	9,197	4,385	47.7	3,629	75.4	13,757	1.496
Verb	5,359	3,763	70.2	1,386	86.8	21,827	4.073
Adverb	583	188	32.2	283	71.6	460	0.789
Total	60,626	32,056	52.9	22,146	77.5	89,388	1.883

a Percentage of keywords mapped to WordNet senses.
b Percentage of cases where *no* mapping could be found.

TABLE 4.23: Precision, recall, and $F_{0.5}$ results of running Verto with the original dictionaries and all of the search strategies.

Lexical category	Mappings		Precision	Recall	$F_{0.5}$
	Correct	Incorrect			
Noun	158	9	0.946	0.272	0.633
Adjective	219	10	0.956	0.336	0.698
Verb	228	51	0.817	0.229	0.540
Adverb	111	10	0.917	0.311	0.660
Total	716	80	0.899	0.277	0.621

The graphs in figures 4.16 to 4.18 on pages 123–125 sum up the precision, recall, and $F_{0.5}$-score results from test runs 1–6. The full-range y-axis makes it easier to compare the result summary of the first six test runs with the results of the last six, shown in figures 4.19 to 4.21 on pages 133–134.

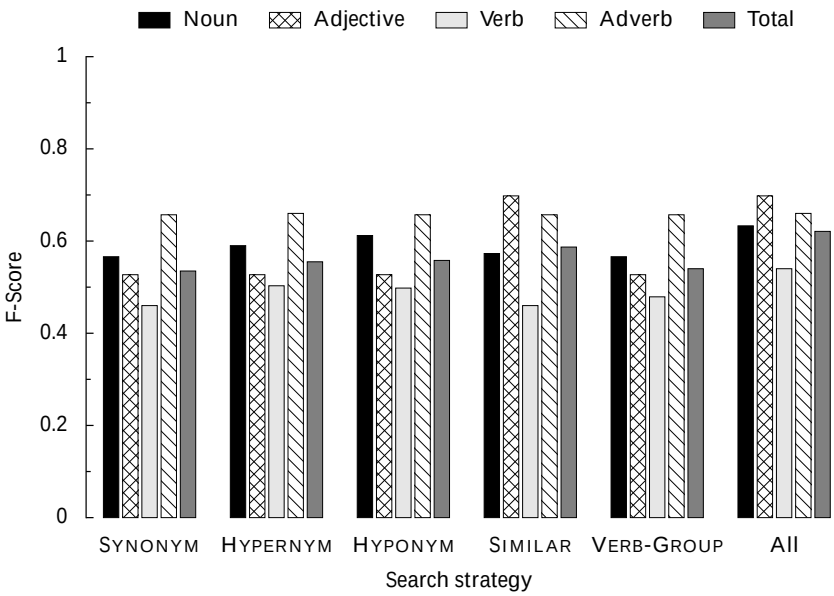


FIGURE 4.18: Summary of $F_{0.5}$ scores through test runs 1–6.

TABLE 4.24: Number of times that no mapping could be found due to missing WordNet entries, with extended dictionaries.

Lexical category	Word not in WordNet	Change (%)
Noun	4,331	-11.95
Adjective	1,002	-15.30
Verb	161	-23.33
Adverb	93	-16.96
Total	5,587	-13.03

TABLE 4.25: Coverage results of running Verto with the extended dictionaries and the SYNONYM strategy.

Lexical category	Keywords	Mapped		No inverse		Mappings	
		N	(%)[a]	*N*	(%)[b]	*N*	Mean
Noun	45,993	25,268	54.9	16,394	79.1	78,857	1.715
Adjective	9,526	4,404	46.2	4,120	80.4	22,022	2.312
Verb	5,435	4,052	74.6	1,222	88.4	31,574	5.809
Adverb	1,394	488	35.0	813	89.7	1,605	1.151
Total	62,348	34,212	54.9	22,549	80.1	134,058	2.747

a Percentage of keywords mapped to WordNet senses.
b Percentage of cases where *no* mapping could be found.

4.10.5 *With the Extended Dictionaries*

During the final six test runs, the original dictionary modules, NORENG and ENGNOR, were replaced by the extended dictionary modules, NORENG_C and ENGNOR_C, respectively.

Because the extended dictionary modules were generated by combining the *Norsk–engelsk stor ordbok* dictionary with the entries obtained by reversing every translation in *Engelsk–norsk stor ordbok*, and vice versa, each extended dictionary is guaranteed to contain a number of entries greater than or equal to the originals they are extensions of. This is in accordance with the observed reduction in the number of words that could not be mapped because no inverse translation could be found, as shown in Table 4.24 on the previous page. The numbers presented in Table 4.24 were constant for all the last test runs.

Test Run 7 (The SYNONYM Strategy)**.** Since this seventh test run starts a sequence of test runs parallel to test runs 1–6, it will serve as the reference case for the following five test runs. As a consequence, for the next five test runs, parenthetic change indicators are relative to the results of this test run, unless otherwise stated.

Even though the results of Test Run 6 presented new maximum coverage values, those values are all surpassed by the results of the seventh test run, presented in Table 4.25, except for the adjective coverage values, which are slightly lower than in Test Run 6 (-1.5 %).

TABLE 4.26: Precision, recall, and $F_{0.5}$ results of running Verto with the extended dictionaries and the SYNONYM strategy.

	Mappings				
Lexical category	Correct	Incorrect	Precision	Recall	$F_{0.5}$
Noun	208	42	0.832	0.262	0.580
Adjective	272	11	0.961	0.224	0.580
Verb	344	67	0.837	0.241	0.560
Adverb	205	22	0.903	0.420	0.734
Total	1,029	142	0.879	0.262	0.598

The average number of mappings for each lexical category was much higher than in the previous test runs, with a total average of mappings per input word of 2.747.

As shown in Table 4.26, all the recall values increased compared to the first run, while they were actually reduced for categories noun and adjective when compared to Test Run 6.

The total $F_{0.5}$ score was higher than in Test Run 1 (+ 0.090), and even slightly higher than in Test Run 6 (+ 0.004).

It should also be noted that the total number of mappings compared to the test set increased with the introduction of the extended dictionaries, as shown in Table 4.26.

Test Run 8 (The SYNONYM and HYPERNYM Strategies)**.** The eighth run used the same MIRROR strategies as Test Run 2, and the same patterns in coverage increase are seen, relative to the foregoing test run. As Table 4.27 on the following page shows, the number of mapped words were increased for all but the adjective category. Consequently, the percentage of words Verto was able to map represented a new maximum (at 56.1 %).

Furthermore, the high average number of mappings per verb (8.030) should be noted.

As shown in Table 4.28 on the next page, precision was reduced, while both the recall and $F_{0.5}$ scores were increased, relative to Test Run 7. The new $F_{0.5}$ score represents a new maximum.

Test Run 9 (The SYNONYM and HYPONYM Strategies)**.** The ninth run replaced the HYPERNYM search strategy with the HYPONYM search strategy.

TABLE 4.27: Coverage results of running Verto with the extended dictionaries and the SYNONYM and HYPERNYM search strategies.

Lexical category	Keywords	Mapped		No inverse		Mappings	
		N	(%)[a]	*N*	(%)[b]	*N*	Mean
Noun	45,993	25,797	56.1	15,865	78.6	90,552	1.969
Adjective	9,526	4,399	46.2	4,125	80.5	22,036	2.313
Verb	5,435	4,260	78.4	1,014	86.3	43,645	8.030
Adverb	1,394	491	35.2	810	89.7	1,609	1.154
Total	62,348	34,947	56.1	21,814	79.6	157,842	3.367

a Percentage of keywords mapped to WordNet senses.
b Percentage of cases where *no* mapping could be found.

TABLE 4.28: Precision, recall, and $F_{0.5}$ results of running Verto with the extended dictionaries and the SYNONYM and HYPERNYM search strategies.

Lexical category	Mappings		Precision	Recall	$F_{0.5}$
	Correct	Incorrect			
Noun	244	44	0.847	0.308	0.627
Adjective	272	11	0.961	0.224	0.580
Verb	435	100	0.813	0.304	0.609
Adverb	206	22	0.904	0.422	0.736
Total	1,157	177	0.867	0.295	0.625

TABLE 4.29: Coverage results of running Verto with the extended dictionaries and the SYNONYM and HYPONYM search strategies.

Lexical category	Keywords	Mapped		No inverse		Mappings	
		N	(%)[a]	*N*	(%)[b]	*N*	Mean
Noun	45,993	25,810	56.1	15,852	78.5	88,070	1.915
Adjective	9,526	4,400	46.2	4,124	80.5	22,028	2.312
Verb	5,435	4,258	78.3	1,016	86.3	38,541	7.091
Adverb	1,394	492	35.3	809	89.7	1,609	1.154
Total	62,348	34,960	56.1	21,801	79.6	150,248	3.118

a Percentage of keywords mapped to WordNet senses.
b Percentage of cases where *no* mapping could be found.

TABLE 4.30: Precision, recall, and $F_{0.5}$ results of running Verto with the extended dictionaries and the SYNONYM and HYPONYM search strategies.

	Mappings				
Lexical category	Correct	Incorrect	Precision	Recall	$F_{0.5}$
Noun	238	45	0.841	0.300	0.618
Adjective	272	11	0.961	0.224	0.580
Verb	387	87	0.816	0.271	0.582
Adverb	205	22	0.903	0.420	0.734
Total	1,102	165	0.870	0.281	0.613

TABLE 4.31: Coverage results of running Verto with the extended dictionaries and the SYNONYM and SIMILAR search strategies.

Lexical		Mapped		No inverse		Mappings	
category	Keywords	*N*	(%)[a]	*N*	(%)[b]	*N*	Mean
Noun	45,993	25,311	55.0	16,351	79.1	79,238	1.723
Adjective	9,526	5,364	56.3	3,160	75.9	35,028	3.677
Verb	5,435	4,052	74.6	1,222	88.4	31,574	5.809
Adverb	1,394	488	35.0	813	89.7	1,605	1.151
Total	62,348	35,215	56.5	21,546	79.4	147,445	3.090

a Percentage of keywords mapped to WordNet senses.
b Percentage of cases where *no* mapping could be found.

Table 4.29 on the facing page shows few surprises. The coverage values were mostly the same as for Test Run 8. However, the total average number of mappings per input word was slightly reduced, mostly caused by a corresponding reduction in the verb category.

As shown in Table 4.30, the overall precision showed a minute increase (+ 0.003), while both the recall and $F_{0.5}$ scores were reduced, relative to Test Run 8.

Test Run 10 (The SYNONYM and SIMILAR Strategies)**.** In the tenth run, the SIMILAR strategy was used, just as in Test Run 4.

As seen in Table 4.31, the strategy gave rise to a new total coverage percentage (of 56.5 %), which represents the maximum so far.

TABLE 4.32: Precision, recall, and $F_{0.5}$ results of running Verto with the extended dictionaries and the SYNONYM and SIMILAR search strategies.

	Mappings				
Lexical category	Correct	Incorrect	Precision	Recall	$F_{0.5}$
Noun	213	42	0.835	0.269	0.587
Adjective	473	30	0.940	0.390	0.734
Verb	344	67	0.837	0.241	0.560
Adverb	205	22	0.903	0.420	0.734
Total	1,235	161	0.885	0.315	0.650

TABLE 4.33: Coverage results of running Verto with the extended dictionaries and the SYNONYM and VERB-GROUP search strategies.

Lexical		Mapped		No inverse		Mappings	
category	Keywords	*N*	(%)[a]	*N*	(%)[b]	*N*	Mean
Noun	45,993	25,268	54.9	16,394	79.1	78,857	1.715
Adjective	9,526	4,404	46.2	4,120	80.4	22,022	2.312
Verb	5,435	4,054	74.6	1,220	88.3	33,537	6.171
Adverb	1,394	488	35.0	813	89.7	1,605	1.151
Total	62,348	34,214	54.9	22,547	80.1	136,021	2.837

a Percentage of keywords mapped to WordNet senses.
b Percentage of cases where *no* mapping could be found.

Table 4.32, shows that both the total precision and recall reached new local—to this partition of the test runs—maximum values (at 0.885 and 0.315, respectively). As a result of the high precision and recall values, the $F_{0.5}$ score reached a new maximum, too (at 0.650).

Test Run 11 (The SYNONYM and VERB-GROUP Strategies)**.** The eleventh test run used the VERB-GROUP strategy, just as Test Run 5. And, just as for Test Run 5, the results—presented in tables 4.33 and 4.34—show few interesting changes, except that the number of average mappings per verb increased—as would be expected, given the strategy's workings—and that the $F_{0.5}$ score showed a minute increase, relative to Test Run 7.

Test Run 12 (The SYNONYM and All Search Strategies)**.** The final run used the extended dictionaries and all of the search strategies.

TABLE 4.34: Precision, recall, and $F_{0.5}$ results of running Verto with the extended dictionaries and the SYNONYM and VERB-GROUP search strategies.

	Mappings				
Lexical category	Correct	Incorrect	Precision	Recall	$F_{0.5}$
Noun	208	42	0.832	0.262	0.580
Adjective	272	11	0.961	0.224	0.580
Verb	364	72	0.835	0.255	0.574
Adverb	205	22	0.903	0.420	0.734
Total	1,049	147	0.877	0.267	0.602

TABLE 4.35: Coverage results of running Verto with the extended dictionaries and the SYNONYM and all of the search strategies.

Lexical		Mapped		No inverse		Mappings	
category	Keywords	*N*	(%)[a]	*N*	(%)[b]	*N*	Mean
Noun	45,993	25,879	56.3	15,783	78.5	99,594	2.165
Adjective	9,526	5,357	56.2	3,167	76.0	35,049	3.679
Verb	5,435	4,264	78.5	1,010	86.3	51,010	9.385
Adverb	1,394	492	35.3	809	89.7	1,610	1.155
Total	62,348	35,992	57.7	20,769	78.8	187,263	4.096

a Percentage of keywords mapped to WordNet senses.
b Percentage of cases where *no* mapping could be found.

TABLE 4.36: Precision, recall, and $F_{0.5}$ results of running Verto with the extended dictionaries and the SYNONYM and all of the search strategies.

	Mappings				
Lexical category	Correct	Incorrect	Precision	Recall	$F_{0.5}$
Noun	273	47	0.853	0.344	0.658
Adjective	473	30	0.940	0.390	0.734
Verb	481	123	0.796	0.337	0.625
Adverb	206	22	0.904	0.422	0.736
Total	1,433	222	0.866	0.365	0.680

Table 4.35 on the preceding page shows how all coverage values increased to reach new maximum levels, except in the adjective category for which Test Run 10 represented the maximum.

Furthermore, the average number of mappings per input word reached its maximum value (of 4.096).

To no surprise, as shown in Table 4.36, the precision level decreased (- 0.013), with respect to Test Run 7, while both the recall and $F_{0.5}$ level reached new maximums (at 0.365 and 0.680, respectively).

Figures 4.19 to 4.21 on pages 133–134 show graphs that respectively sum up the precision, recall, and $F_{0.5}$-score results in test runs 7–12.

The bar graphs in figures 4.16 to 4.18 and 4.19 to 4.21 emphasize differences between the different lexical categories (and the total) within each test run. They also make it easy to get an impression of the sizes involved. However, the bar graphs are less suited to emphasize *change* between each test run. Therefore, the line graphs in Figure 4.22 are included to emphasize how the precision and recall—and also $F_{0.5}$—results change from test run to test run.

4.10.6 Ordnett

As mentioned above, the resulting mappings from the experiments were used to create Ordnett, a novel Norwegian lexical-semantic resource that maps Norwegian words to WordNet senses. Because of Verto's mapping strategy, Ordnett inherits WordNet's rich number of semantic relations.

Figure 4.23 on page 136 shows a small segment of Ordnett. The figure shows the mappings—designated by dashed arrows—from (base forms

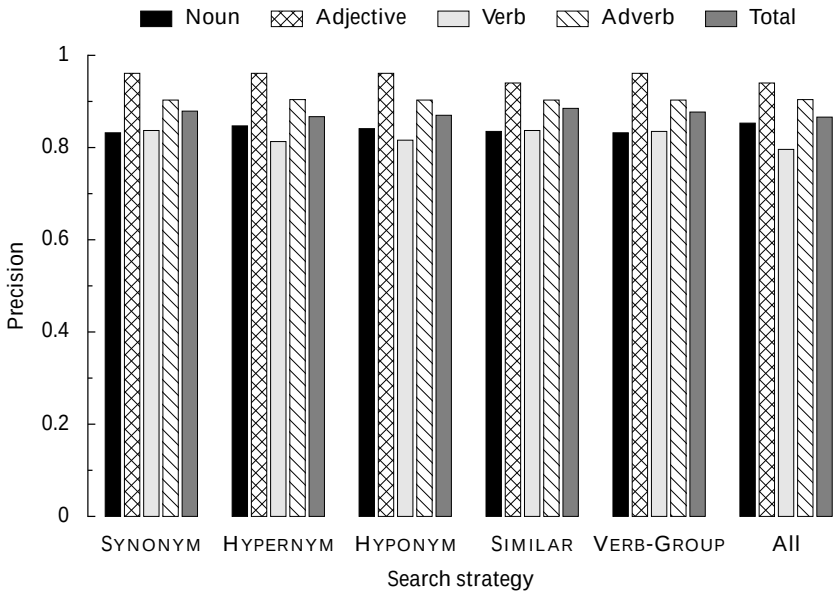


FIGURE 4.19: Summary of precision scores through test runs 7–12.

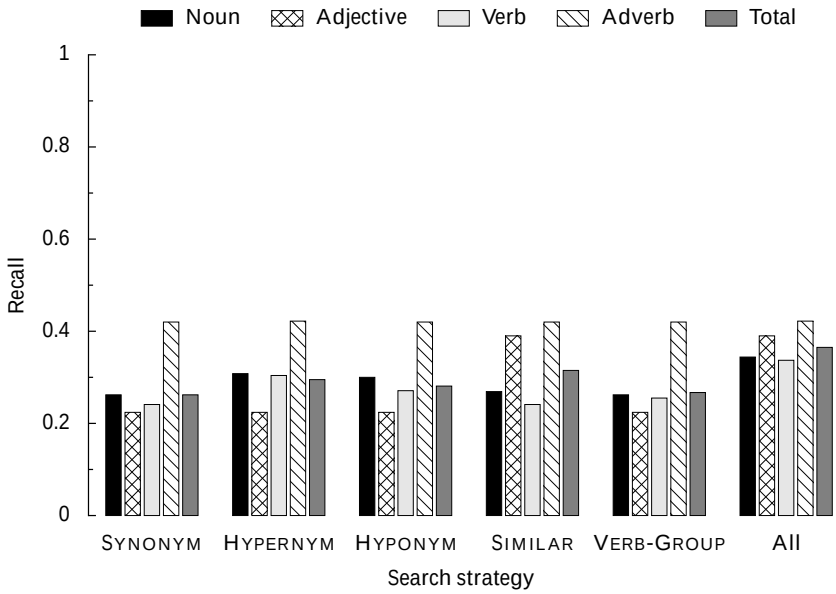


FIGURE 4.20: Summary of recall scores through test runs 7–12.

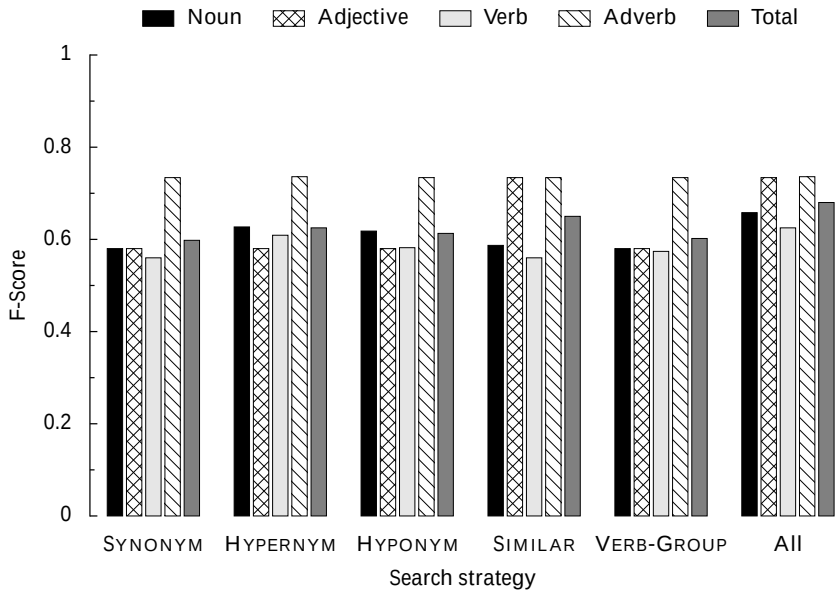


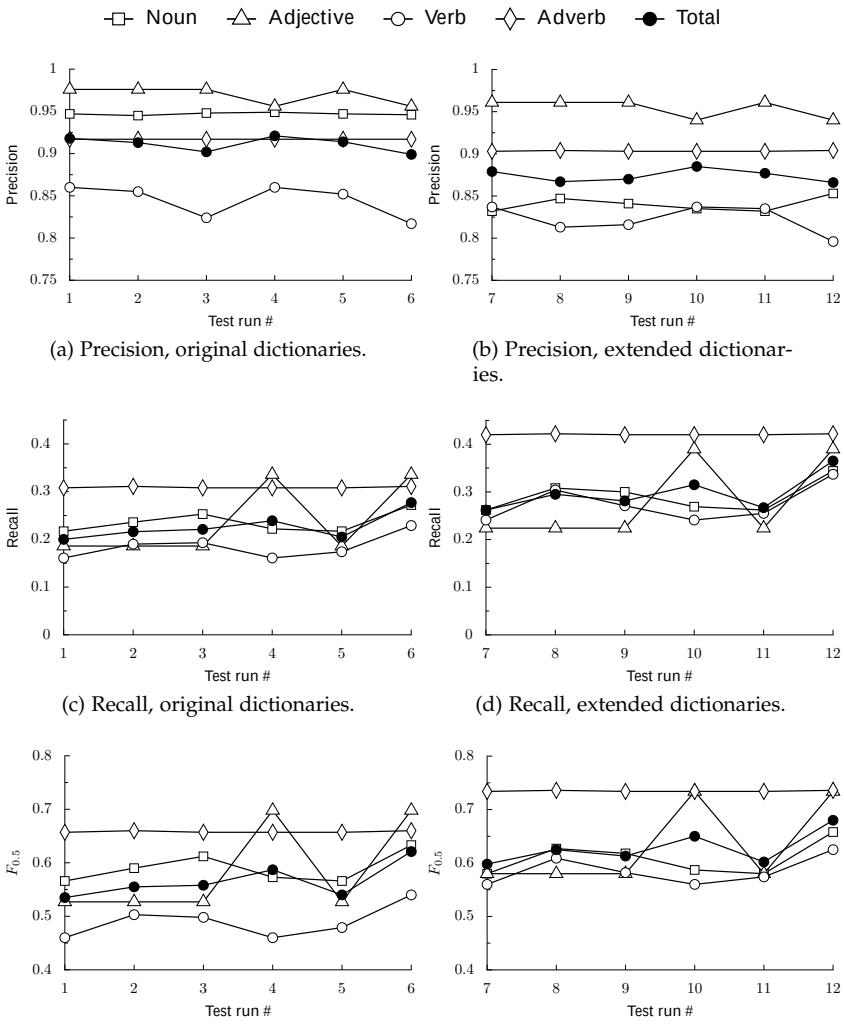
Figure 4.21: Summary of $F_{0.5}$ scores through test runs 7–12.

of) Norwegian words to synsets in WordNet. The figure also shows both meronymy (part of)—designated by arrows with diamond-shaped arrowheads—and hypernymy (a kind of) relations.

Regarding Figure 4.23, it should be noted that Verto actually finds mappings from words to WordNet *senses*, not synsets. However, to make the figure easier to comprehend, I let the mappings target the synsets that the senses belong to. Also note that the figure is non-exhaustive in that the shown synsets are related to many more synsets than can be shown here, and a few synsets along the hypernymy paths have been left out, because of space concerns, and replaced by ellipses.

4.11 Analysis and Discussion

As shown by the results presented in Section 4.10.1, Verto—the mapping framework—generally offers a high level of precision. Figure 4.22 provides a summary of how the precision, recall, and $F_{0.5}$-measure changed during the twelve test runs.



(a) Precision, original dictionaries.

(b) Precision, extended dictionaries.

(c) Recall, original dictionaries.

(d) Recall, extended dictionaries.

(e) $F_{0.5}$, original dictionaries.

(f) $F_{0.5}$, extended dictionaries.

FIGURE 4.22: The change patterns in the results from test runs 1–12.

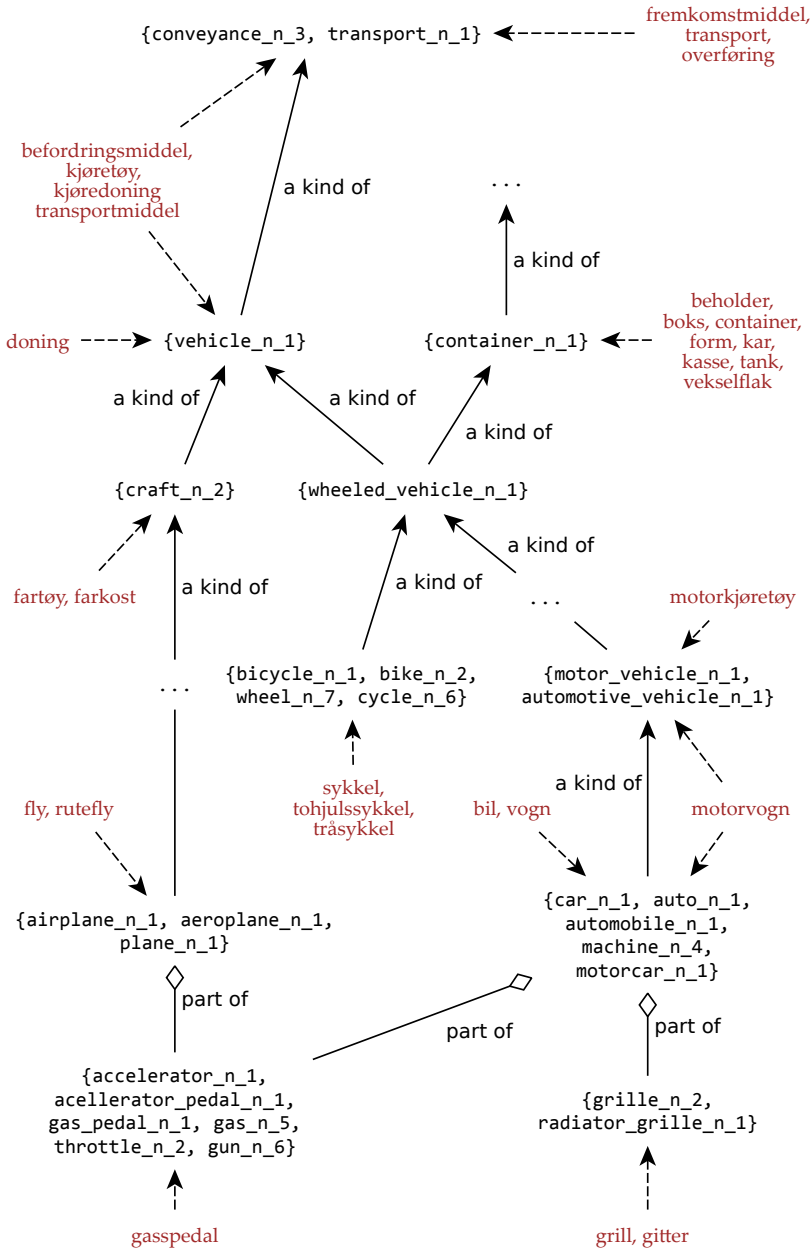


FIGURE 4.23: A small, non-exhaustive segment of the Ordnett resource, showing mappings from Norwegian to WordNet synsets in addition to both meronymy and hypernymy relations.

The highest precision value—averaged over all the lexical categories—at 0.921, was achieved in Test Run 4, featuring the original dictionaries. However, in the same test run, the total recall value was 0.239, as shown in Table 4.19.

Even though the F-measure was weighted with $\beta = 0.5$—preferring precision over recall—the highest average $F_{0.5}$ score with the original dictionaries was achieved in Test Run 6 at 0.621; see Figure 4.22e and Table 4.23. Globally, the highest average $F_{0.5}$ score was achieved in Test Run 12, as shown in Figure 4.22f and Table 4.36.

During test runs 7–12 the overall recall generally increased, reaching a maximum at 0.365 during Test Run 12, as shown in Table 4.36. However, during the same run, the precision level was 0.866, while the $F_{0.5}$-measure was 0.680.

During the test runs, the number of words Verto found a warranted mapping for increased from 29,826 (49.2 %), in Test Run 1, to 35,992 (57.7 %), in Test Run 12, of the total 62,348 words given as input.

All in all, the tests show that the Verb-Group search strategy made insignificant contributions to both the precision and recall levels. The Hyponym and Hypernym search strategies on the other hand, both were able to increase the recall level slightly at the expected cost of a slightly decreased level of precision. The most surprising result was probably the Similar search strategy's effect on the adjective recall level; its contribution caused the total recall level to increase more than both the Hyponym and Hypernym search strategies did.

Through all the tests, the adverb category's precision, recall, and $F_{0.5}$ levels were unaffected by all of the applied search strategies. However, the adverb category's levels were higher than the total levels in each test. In contrast, the precision, recall, and $F_{0.5}$ levels of the verb category generally were lower than the total levels.

Furthermore, it seems that the combination of all the search strategies, in test runs 6 and 12, caused new maximum $F_{0.5}$ levels to be reached for each of the lexical strategies. This suggests that applying the search strategies in combination causes a synergy effect; the combined application has greater effect per affected lexical class than applying them separately.

The results also show that extending the dictionary resources made available to Verto made a great impact on both the coverage, precision, and recall levels. Additionally, the average number of mappings per input word increased significantly by this extension.

In Section 4.11.1 attempts were made at discovering trends in both precision and recall levels that might correlate with the input words' usage frequencies.

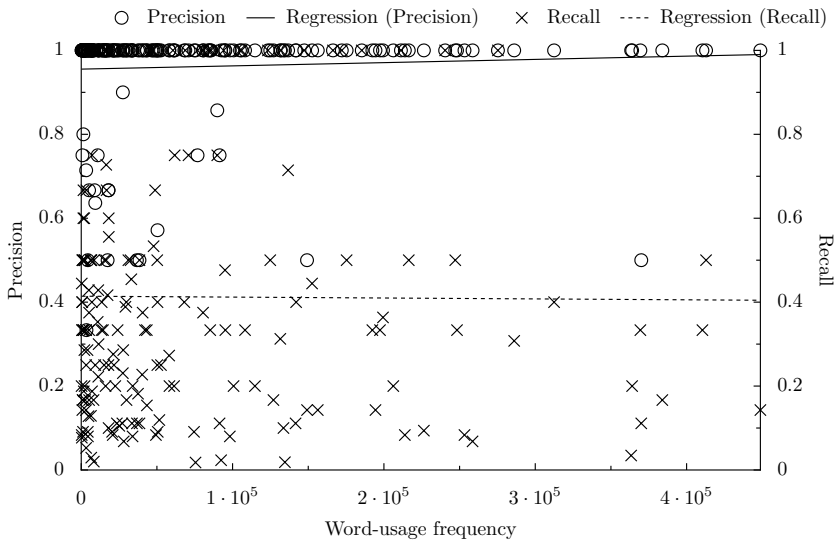


FIGURE 4.24: Scatter graph of precision and recall scores for each input word against its usage frequency, with regression lines.

The correlation r will always be $-1 \leq r \leq 1$, and values of r close to 0 indicate a very weak relationship between the variables. Therefore, the scatter graph in Figure 4.24 and the very low correlation between word-usage frequencies and both precision ($r = 0.061$) and recall ($r = -0.007$) show that there is no significant straight-line relationship between the word-usage frequencies of the input words and the precision and recall of the system.

4.11.1 Word-Usage Frequencies

To see whether the common usage frequency of input words affect the results of the Verto method, I prepared a scatter graph. The scatter graph, based on values from Test Run 1, is shown in Figure 4.24, and contains scatterplots of both precision and recall values for each word against the word's usage frequency. The word frequencies were gathered from the Oslo Corpus of Tagged Norwegian Texts (Text Laboratory 2008b), and were based on Norwegian newspaper articles in both Nynorsk and Bokmål. The word-frequency list for Bokmål was based

on 14,453,053 occurrences and contains 460,416 unique words, while the word-frequency list for Nynorsk was based on 3,714,453 occurrences and contains 169,028 unique words.

To see whether there was any straight line relationship between the word usage frequency and either the precision or recall values, the correlation between the usage frequency and the two measures was calculated. The formula for correlation, r is

$$r = \frac{\sum \left(\frac{x-\overline{x}}{s_x}\right)\left(\frac{y-\overline{y}}{s_y}\right)}{n-1} \tag{4.32}$$

where x and y represent two variables for n observations. For example, given that the correlation between word-usage frequencies and precision levels is calculated, x represents the word usage frequency and y represents the precision. The symbols $\overline{x}$ and $\overline{y}$ represent the mean for these x and y, respectively, while s_x and s_y represent their standard deviations. The standard deviation s_x is

$$s_x = \sqrt{\frac{\sum (x-\overline{x})^2}{n-1}} \tag{4.33}$$

and s_y is calculated equally; just replace x by y.

The correlation between the word-usage frequencies and precision levels is 0.061, while the correlation between the word-usage frequencies and recall levels is -0.007.

In an attempt to emphasize trends if there were any, Figure 4.24 also contains simple least-squares regression lines for the precision and recall observations.

4.11.2 *Misaligned Coverage*

In a few cases, problems were caused by *Norsk–engelsk stor ordbok*, *Engelsk–norsk stor ordbok*, and WordNet covering disjoint sets of senses for certain words. This can be seen as a problem with misaligned coverage between *Norsk–engelsk stor ordbok*, *Engelsk–norsk stor ordbok*, and WordNet.

For example, one of the translations of the Norwegian verb «*krote*» (to deck something with scrolls, often by carving) in *Norsk–engelsk stor ordbok* is the English verb *scroll*. However, in WordNet *scroll* as a verb has only one sense, and that sense is "to move through text or graphics in order to display parts that do not fit on the screen." In this case, there is no way the inverse-translation principle can be used to find

the warranted mapping, because WordNet does not cover the senses referred to by *Norsk–engelsk stor ordbok/Engelsk–norsk stor ordbok*.

The same problem has been observed for a few nouns. For example, the Norwegian noun *vift* (a puff or a whiff) is translated into the English noun *waft*, while in WordNet the only sense of *waft* as a noun is "a long flag."

After the dictionaries were combined, the precision decreased, while the recall increased. This was no big surprise, as there is usually a tradeoff between the two measures.

For example, when the dictionaries were combined, new possible translations were introduced, and a problem similar to the one mentioned above became visible. For example, the EngNor module provides a translation from "poppet" to *«skatt»*, while the introduction of the combined dictionary resource NorEng$_C$ therefore adds a new translation from *«skatt»* to "poppet". The problem arises when "poppet" is found in WordNet, because the only sense of "poppet" in WordNet is "a mushroom-shaped valve that rises perpendicular from its seat; commonly used in internal-combustion engines"; a sense not warranted by *«skatt»*.

4.11.3 Ordnett

The mappings found by Verto were used to create Ordnett, a lexical-semantic resource that maps Norwegian words to WordNet senses. Thereby, we obtain access to a rich lexical-semantic resource for Norwegian, because Ordnett inherits the rich number of semantic relations defined in WordNet.

This section will discuss some of the observations made with regard to the creation and possible applications of Ordnett.

Cross-Synset Mappings

As shown by the mappings of the Norwegian words *«motorvogn»*, *«befordringsmiddel»*, *«kjøretøy»*, *«kjøredoning»*, and *«transportmiddel»* in Figure 4.23 on page 136, the rather fine grained distinction made between some synsets in WordNet does not necessarily fit the meaning of the mapped words perfectly. From the examples in Figure 4.23, it seems that some Norwegian words, that are synonymous in some contexts, risk getting mapped to two (or, perhaps, more) *closely related* synsets in WordNet. It appears that these cross-synset mappings can be traced back to the dictionary resources. For example *«motorvogn»* can be used about any kind of motorized car or vehicle, and *Norsk–*

engelsk stor ordbok therefore provides translations to both "motor car" and "motor vehicle"; something that eventually lead Verto to accept both mappings, even though they target different synsets.

On the other hand, one might also argue that these mappings are a consequence of WordNet being defined with so fine-grained distinctions that what differentiates two synsets is sometimes hard to define.

However, deciding whether a cross-synset mapping is caused by such a combination of source-language synonymy and fine-grained synset distinctions, or simply reflect *different* meanings of a homograph, is important. One way to decide could, for example, be to only consider cross-synset mappings that are *one* hypernym/hyponym step apart as equal, or similar, in meaning. Such a decision is based on a view that true homographs would be more distinct in meaning.

Nonetheless, in practical use of Ordnett one could handle such cross-synset mappings in several ways, depending on the application. One way could be to merge all features that hold for each involved synset, to create a proxy synset. Another way could be to always choose either the most or the least general synset. A third way could be to choose the synset that fits, for example, the current context in the application that uses Ordnett.

Possible Applications

Generally, it should be possible to utilize the lexical semantic resource Ordnett in applications that process Norwegian language within most, if not all, of the NLP areas described in Section 2.1.2. Even though Chapter 5 will show how Ordnett can be applied to the field of text interpretation and open-domain question answering (open-domain QA), because most applications in those areas are quite complex, going into detail about how it can be used in each of those is outside the scope of this dissertation. Generally, Ordnett can be used as a Norwegian general ontology that has all of the semantic relationships defined in WordNet. Furthermore, because the mapping of Norwegian words serves as an interface to WordNet, Ordnett can also be used together with other resources that interface WordNet to extend or complement it.

On the other hand, one more light-weighted application of Ordnett, that I would like to mention, is within on-line ad placement or product suggestions. For example, because of its rich number of semantic relationships, inherited from WordNet, Ordnett could very well be used for presenting adverts to users based on search terms used in Web queries. This could be done by utilizing Ordnett's (inherited) hypernymy and

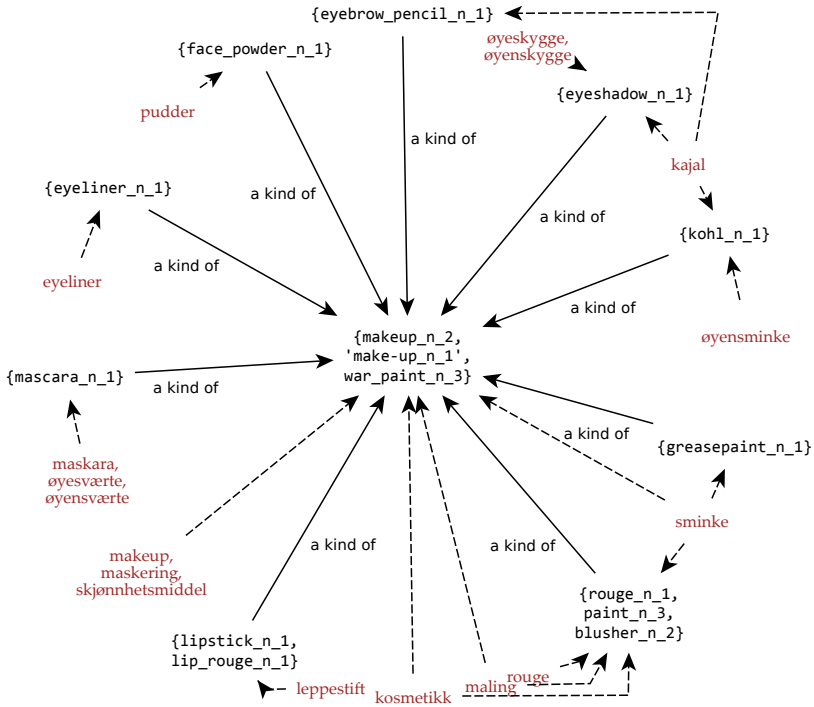


FIGURE 4.25: Ordnett mappings from Norwegian to the WordNet synset containing `makeup_n_2` and its hyponyms.

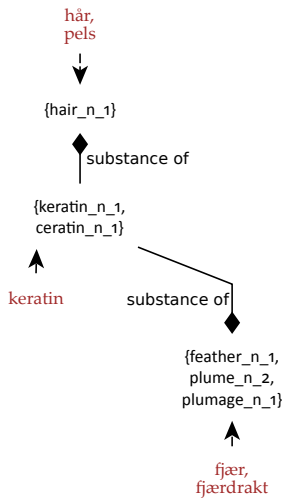


FIGURE 4.26: Ordnett mappings from Norwegian to the WordNet synset containing `hair_n_1` and its substance meronyms.

meronymy relations. Looking back at Figure 4.23 on page 136, if a user, for example, searches for «*gasspedal*», a system could use the meronymy information in the lexical-semantic resource to recognize that accelerators are part of automobiles. Based on that knowledge, the system could present adverts for cars or other car parts to the user. Likewise, the system could use hypernymy information to present users with adverts for different kinds of makeup if the user searches for a particular kind of makeup, for example «*leppestift*» or «*øyensværte*», as shown in Figure 4.25. The synsets {`lip-gloss_n_1`} and {`blackface_n_1`}—both hyponyms of {`makeup_n_2`, `make-up_n_1`, `war_paint_n_3`}—were not included in the figure, because no mappings from Norwegian targeted those.

Example Semantic Relations

To provide a better impression of the richness of semantic information that Ordnett gains access to through WordNet, figures 4.26 and 4.27 show a couple of semantic relations not yet presented.

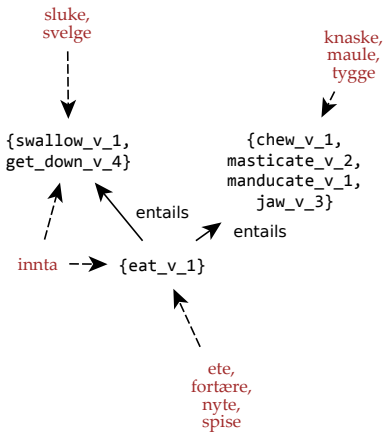


FIGURE 4.27: Ordnett mappings from Norwegian to the WordNet synset containing `eat_v_1` and actions it entails.

The segment of Ordnett presented in Figure 4.26 shows a kind of meronymy called substance meronymy, designated by the arrows with black diamond heads. The substance meronym relations in the figure signify that `ceratin_n_1` is a substance in both `hair_n_1` and `feather_n_1`. Or, in Norwegian, that «*keratin*» is part of both «*hår*» and «*fjær*».

Figure 4.27 shows the entailment relationship in WordNet that is defined for pairs of verb synsets. For example, the Ordnett segment in the figure shows that to `eat_v_1` something entails that one must both `chew_v_1` and `swallow_v_1`. Even though, the order of the chewing and swallowing is not fixed, the information can still be useful for NLP purposes.

4.11.4 Comparison with Semantic Mirrors

Dyvik's semantic mirrors method is a way to derive lexical-semantic resources from translational data, typically parallel corpora.

The method is generally based on an assumption that sense distinctions in the source language are mirrored into the target language.

In particular, the semantic mirrors method is based on the following assumptions (Dyvik 2005)[10]:

(D1) Semantically closely related words tend to have strongly overlapping sets of translations.

(D2) Words with wide meanings tend to have a higher number of translations than words with narrow meanings.

(D3) If a word *a* is a hyponym of a word *b* (such as tasty of good, for example), then the possible translations of *a* will probably be a subset of the possible translations of *b*.

(D4) Contrastive ambiguity, i.e., ambiguity between two unrelated senses of a word, such as the two senses of the English noun band ('orchestra' and 'piece of tape'), tends to be a historically accidental and idiosyncratic property of individual words. Hence we don't expect to find instances of the same contrastive ambiguity replicated by other words in the language or by words in other languages. (More precisely, we should talk about ambiguous phonological/graphic words here, since such ambiguity is normally analysed as homonymy and hence as involving two lemmas.)

(D5) Words with unrelated meanings will not share translations into another language, except in cases where the shared (graphic/phonological) word is contrastively ambiguous between the two unrelated meanings. By assumption (D4) there should then be at most one such shared word.

The semantic mirrors method refers to the set of possible translations—based on word-alignment occurrences in parallel corpora—of a word as the word's first *t*-image. Hence, a word's first *t*-image is the set of all observed translations of the word in the parallel target-language text, while all the first *t*-images of those words back into the source-language text are called the original word's inverse *t*-image.

Furthermore, the semantic mirrors method works by comparing how first *t*-images and inverse *t*-images overlap and form disjoint subsets. By exploiting the information implicitly expressed by the different subsets,

10 The labels have been changed by me. Each of the quoted assumptions are identified by a label (D*n*), where D refers to the author Dyvik and *n* refers to the label used by Dyvik (2005).

the method is able to both separate out individual senses of each word and to derive different semantic relations among them.

The method presented by me, as implemented in Verto, is related to the semantic mirrors method because both methods take advantage of the semantic relations expressed by translational relations. The semantic mirrors method creates a lexical-semantic resource by mining the implicit semantic relations expressed by word-alignments of parallel corpora, while my method creates a lexical-semantic resource by combining the translational relations expressed in dictionaries with the explicit semantic relations expressed by WordNet.

4.11.5 Comparison with Other Approaches

As mentioned in Section 2.4.1, Rigau and Agirre (1995) used a method based on the conceptual density measure for disambiguating mappings from a French-English to WordNet. First of all, it should be noted that the method they presented only deals with nouns, and for each translation they only want to find a single representative WordNet sense. Furthermore, the performance of both their and my method largely depends on the dictionary resources they use. Therefore, comparing Rigau and Agirre's results with the results presented herein might not be appropriate. This is especially true for a comparison of the coverage of the methods, because the coverage is measured as a percentage of all possible translations in the dictionary resources. Nonetheless, Rigau and Agirre report a coverage of 47 % and a precision of 91 %.[11]

In comparison, focusing on nouns only, the corresponding results from Verto are shown in Table 4.37 on the facing page. As can easily be seen, by focusing on maximization of precision Verto achieved 50.3 % coverage and at the same time 94.9 % precision, both results slightly better than those reported by Rigau and Agirre. Optimizing for maximum recall, $F_{0.5}$, or coverage, on the other hand, yields a significant improvement in coverage (57.7 %), at the cost of lowered precision (85.3 %).

11 Rigau and Agirre (1995) do not state their average coverage and precision explicitly. However, they present coverage numbers, distributed over the following categories of translation cases (percentage of the total number of attempted translations): (i) single word with only a single sense in WordNet (2,172 words, 14 %); (ii) (they call this complex translations) multiple words where one of the translations has only a single sense in WordNet (2,947 words, 19 %); (iii) more than one possible translation (723 words, 5 %); and (iv) a cue in French is given provided by the dictionary entry (1,399 words, 9 %). These numbers sum up to a total coverage of 7,241 words, or 47 %. They report the following precision measures: Category (i) 100 %, Category (ii) 88 %, Category (iii) 93 %, and Category (iv) 83 %. Therefore, a weighted mean value for the precision should yield 91 %.

Table 4.37: Verto's precision, recall, and coverage results for mapping of *nouns*.

Maximum	Precision	Recall	$F_{0.5}$	Coverage	Test Run
Precision	**0.949**	0.222	0.573	22,888 (50.3 %)	4
Recall	0.853	**0.344**	0.658	35,992 (57.7 %)	12
$F_{0.5}$	0.853	0.344	**0.658**	35,992 (57.7 %)	12
Coverage	0.853	0.344	0.658	**35,992 (57.7 %)**	12

Using different statistical analyses to evaluate the quality of the broad spectrum of methods for mapping nouns to WordNet 1.5 presented by Atserias et al. (1997), Farreres et al. (2002) manage to present detailed precision and coverage statistics for the methods used to build the Spanish WordNet. Farreres et al. present detailed results, showing the costs of trading coverage for increased precision and vice versa. Their highest reported level of precision is 93.28 %, which is close to—but lower than—Verto's maximum, with an according coverage of only 5.70 %. They also report a configuration that yields a coverage of 44.50 % with a precision of 91.35 %. However, their next step towards increased coverage yields a coverage of 71.80 % with a precision of 90.97 %.

The comparison of Verto's results against the results reported by Rigau and Agirre (1995) and Atserias et al. (1997), through Farreres et al. (2002), indicates—at least for nouns—that Verto's performance is very competitive when optimizing for precision and slightly less competitive when optimizing for coverage.

Because Atserias et al. (1997)'s "variant criterion", shown in Figure 2.6a in Section 2.4.3, exploits a relationship between the synonymy within synsets and the translation relation to determine which mappings are probably correct, the method could at first resemble parts of my method (described in this chapter). However, upon closer inspection it can be seen that the variant criterion is merely hinting at one of the driving forces behind my method.

First of all, Atserias et al. state that a Spanish word is mapped to a synset if two or more of the synonyms in the synset "*have only one* translation to the same Spanish word" (emphasis added by me). Secondly, by exploiting the synonymy relation, their criterion is only using intrasynset information. The method presented in Chapter 3, on

the other hand, is not constrained to English words that only have a single translation into what originally constitutes the source language. Furthermore, compared to the variant criterion, my method uses the synonymy within synsets merely as a stepping stone, since it also applies a variety of search strategies that utilize a variety of semantic relations defined between synsets in WordNet.

—Successfully teaching a computer to do natural language understanding in open domain is one of the major unsolved problems in artificial intelligence.

Grunfeld and Kwok (2006)

5 Open-Domain Natural Language Understanding for Norwegian

This chapter shows how the lexical-semantic resource Ordnett can be used in TUClopedia, an open-domain question answering (open-domain QA) system that automatically extracts and acquires knowledge from Norwegian encyclopedic articles and uses the acquired knowledge to answer questions formulated in natural language by its users.

5.1 Motivation

The encyclopedic articles to be parsed and interpreted by TUClopedia constitute *Store norske leksikon* (Henriksen 2003), a Norwegian encyclopedia that contains information on all branches of knowledge.

1 Because *Store norske leksikon* is a general encyclopedia, which means that it is not constrained to any specific domain of knowledge, it amounts to an open-domain knowledge source.

There are also a couple of other reasons that we constrain our research to encyclopedic texts:

2 The information an encyclopedia provides is *in principle* correct and generally represents grammatically correct and concise written language.

3 We strongly believe that a traditional encyclopedia will benefit from the transformation into a Natural Language Understanding (NLU) system.

As Information Retrieval (IR) technology, such as the Google Web Search[1] engine, becomes part of people's everyday life—taking the

1 Google Web Search, `http://www.google.com/`. Accessed July 22, 2007.

global, social, and democratic aspects of the current digital divide (Norris 2001) into account—some inherent limitations of traditional, printed encyclopedias become obvious.

Motivation

For example, the traditional organization of encyclopedia articles, where the articles are alphabetized according to a single descriptive topic word or phrase that identifies the topic or theme of each article, is well suited for the book medium. It might even be argued that the traditional organization probably is *optimal*, given the constraints of the book medium, and that the topic words are wisely chosen. And, even though some encyclopedias are accompanied by index volumes that provide article references based on complementary—but still descriptive—terms, use of the traditional, printed encyclopedias still becomes cumbersome, compared to querying on-line databases.

So far, however, when traditional encyclopedias have been made available electronically—see for example Encyclopædia Britannica Online[2], or the on-line version of *Store norske leksikon*[3]—they have been made alphabetically browsable by subject, by topic, or by index words. They also offer search features, so that users may search for articles containing specific words or phrases. Even the newer Wikipedia[4] offers the same article access methods as just described; that is, through browsing and keyword searching.[5] Nonetheless, even when augmenting encyclopedias, with indexes for traditional and keyword searches for electronic encyclopedias, there are still some tasks that are rather cumbersome to perform.

For example, it is still difficult to find answers to questions that require reading multiple articles in the encyclopedia because the answers are only partially provided by single articles. An example of this is a question like "what are the names of the three highest mountains in Norway?" Provided that one has access to an on-line version of an encyclopedia enriched with an IR interface, but does not know the name of any mountain in Norway, one would probably search for the terms "mountain", and "Norway". Furthermore, the search engine would probably return several candidate articles, of which only a few really contain pieces of the answer (unless, of course, it finds a single article that contains a list of the highest mountains in Norway). Consequently, one would not have much other choice than to start reading all

2 Encyclopædia Britannica Online, `http://www.britannica.com/`. Accessed July 22, 2007.

3 The on-line version of Store norske leksikon, `http://www.snl.no/`. Accessed July 22, 2007.

4 Wikipedia, `http://www.wikipedia.org/`. Accessed July 22, 2007.

5 Although it should be noted that Powerset, mentioned in Chapter 1, currently provides a question-answering (QA) interface for a limited number of Wikipedia articles.

the returned articles, one by one, and combine the adequate pieces of information until the question has been confidently answered.

Generally, to use a modern IR system—as found on the Web today—to find answers to a question, one must first spend time to transform the question into a query with keywords that increase the chances of receiving relevant candidate documents. Then, if the IR system finds any relevant documents, one must spend time analyzing the documents, looking for relevant passages that answer the question. In short, as IR technology is becoming part of everyday life, users want to spend less time translating their questions into keyword-based queries and analyzing candidate documents for answers, and more time utilizing the answers they seek in the first place.

Therefore, considering the above factors, an interesting test for the Ordnett resource—produced with the method presented in Chapter 4—will be to see how it can be utilized for NLU and QA in the open domain that the texts from the general Norwegian encyclopedia (Henriksen 2003) constitute.

Furthermore, as noted by Paşca (2003), the design and implementation of open-domain QA systems raise several questions, including (1) "What kind of unified model could be envisaged to represent information encoded in open-domain questions?" and (2) "What types of natural language processing techniques are best suited to find answer strings within documents?"

Based on earlier experiences with successfully applying *The Understanding Computer* (TUC) to several narrow domains we want to explore how the NLU system can be applied to open-domain QA and how TUC must be modified to accomplish this. By putting Ordnett to test in an open-domain QA system, we also hope to gain insight that can contribute to answer the above questions posed by Paşca.

However, TUC is a NLU system that performs deep analyses that include semantic type checking of candidate interpretations during text parsing and therefore relies heavily on the availability of a semantic resource in the form of a semantic network, or ontology.

Below, I will show how the TUC system was extended and modified to perform open-domain QA for Norwegian texts, utilizing Ordnett and WordNet as its main lexical semantic resources.

5.2 Scaling up TUC

The previous TUC-based projects have considered rather narrow domains of discussion. For example, some indicative numbers for the BusTUC system—420 nouns, 150 verbs, 165 adjectives, 60 prepositions

and about 4000 entries in the semantic network (Amble 2000)—show that this is reflected both in the size of the lexicons and the semantic network.

Semistructured Resources

A much larger lexicon and a larger semantic network will be needed in order to process and interpret a complete encyclopedia. In order to meet the new requirements, TUClopedia—the project's adaption of TUC—was expanded in several ways. First of all, the Norwegian lexicon has been considerably extended by using *Norsk komputasjonelt leksikon* (NorKompLeks) (Nordgård 1998), a computational lexicon that contains both morphological and syntactic information for Norwegian. Additionally, the lexical semantic network WordNet 2.1 (Harabagiu et al. 1999) is made available to TUC in order to increase TUC's semantic knowledge. As mentioned in Section 4.1.2, WordNet 2.1 contains 117,097 nouns, 11,488 verbs, 22,141 adjectives and 4,601 adverbs.

Because of the quite specialized, earlier applications of TUC, during several years of development the grammar has slowly become "tainted" with pragmatic rules—rules that have been very adequate and successful in these narrow domains, but with little or no general applicability. The development of TUClopedia will take part in an ongoing effort to develop a new grammar for TUC, using the same Context-Sensitive-Categorial-Attribute-Logic (ConSensiCAL) formalism as the previous grammar, but based on a reference grammar for Norwegian (Faarlund et al. 1997). Another reason for writing a new grammar is the assumption that encyclopedias are written with correct spelling and use of the language, while for example BusTUC has to cope with very oral formulations.

5.3 Semistructured Resources

The documents used as source material for the system were articles from the general, Norwegian encyclopedia *Store norske leksikon* (Henriksen 2003). Just as for the Norwegian–English and English–Norwegian dictionary resources (Haslerud and Henriksen 2003) described in Section 4.1.1, the format was semistructured, using Extensible Markup Language (XML), and the provided files contained all the structural information used to typeset and actually print the books they represent.

The XML documents contain tags marking fragments of the text as, for example, keywords, biographic names, and titles of works. Some of the title-of-work tags also contain information about what kind of work it is, like 'literature', 'film', 'music', etc.

During preprocessing of the articles, such named entities are extracted and stored in an index that maps from the named entity to the

article where it was tagged. Later, when the articles are parsed, this knowledge solves several inherently difficult problems.

For example, Hull and Gomez (1999) report that their system had difficulties with recognizing titles in running text. The system described in this dissertation applies a multiple-pattern approximate sequence matching algorithm, described in Section 5.5, that uses the information about the already extracted and indexed named entities during a multitagging stage.

5.4 System Overview

Figure 5.1 on the next page shows the main components of the implemented TUClopedia system. The main NLU part of the system is LexTUC, a version of TUC adapted to TUClopedia. The different resources (denoted by drum symbols—or cylinders—in Figure 5.1) used and created by the system are:

- *Bilingual dictionary,* contains *Norsk–engelsk stor ordbok* and *Engelsk–norsk stor ordbok* (Haslerud and Henriksen 2003), described in Section 4.1.1.
- *Lexicon,* is NorKompLeks (Nordgård 1998), described in Section 3.2.
- *Ordnett,* is the resource—created with the framework described in Section 4.9—that maps Norwegian words and phrases to concepts in WordNet.
- *Encyclopedia,* is *Store norske leksikon* (Henriksen 2003) in XML format, stored in an Extensible Markup Language database (XML database)[6].
- *Grammar,* is LexTUC's grammar developed according to the ConSensiCAL grammar formalism described in Section 2.7.1. The grammar rules are based on principles from *Norsk referansegrammatikk* by Faarlund et al. (1997) and *Norsk generativ syntaks* by Åfarli and Eide (2003).
- *Semantic network,* refers to LexTUC's manually developed internal semantic network, which is just an interface to WordNet.
- *WordNet,* is the Princeton WordNet (Miller and Fellbaum 2007).
- *Onomasticon,* is an index containing a list of everything that was XML tagged as some kind of named entity in *Store norske leksikon*.
- *Knowledge base,* contains rules for reasoning and all the knowledge acquired from encyclopedia articles.

6 Oracle's Berkeley DB XML, version 2.4.16.

FIGURE 5.1: TUClopedia's architecture.

5.4.1 Preparations

Prior to operational use, the system parses the semi-structural encyclopedia articles and stores them in a relational database ("Encyclopedia", in the figure) in order to provide easier random access to parts of documents.

After the encyclopedia articles have been stored in the database, they are processed by the

- *Named-entity extractor*, responsible for building the Onomasticon—an index of named entities, based on structural information in the XML source documents. This component runs through all the encyclopedia articles prior to the knowledge-acquisition and QA phase (indicated with dashed-line arrows in Figure 5.1).

The main operating loop (denoted by dotted-line arrows in Figure 5.1) starts with an encyclopedia article during the knowledge acquisition phase, and with a question posed by a user during the QA phase. With this main operating loop as a starting point, the different processing components of TUClopedia will be described below.

5.4.2 Preprocessing and Lexical Analysis

The entry point of TUClopedia is the preprocessing module that is responsible for lexical analysis of the input. This section will present the processing modules of the system (denoted by boxes with rounded corners in Figure 5.1) that are mainly concerned with this task:

- *Lexical analyzer*, responsible for tokenizing and tagging the input to the system. The result is a word graph where each part of the input is tagged with—if necessary, multiple—tags identifying different aspects of the both simplex words and phrases. Thus, the lexical analyzer is a multitagger.

  The aspects identified in the word graph include the constituent's lexical category, morphology, recognized named entities, and possible matching representations in WordNet. To identify matching representations in WordNet, the component utilizes the Ordnett Norwegian–WordNet mapping resource produced by using the method presented in Chapter 4.

  The system delegates parts of its responsibilities to the Sentence-boundary detector, the Translation framework, and the Named-entity recognizer. This component is also responsible for communicating with the database server that contains the encyclopedia articles.

- *Translation framework,* used by the Lexical analyzer to find mappings from Norwegian words and phrases to concepts in WordNet if they were not found in the existing Ordnett resource. The component uses information available in the Bilingual dictionary and the Lexicon.
- *Named-entity recognizer,* used by the Lexical analyzer to detect and tag known named entities. The component uses information available in the Bilingual dictionary and the Lexicon.
- *Sentence-boundary detector,* used by the Lexical analyzer to detect and tag the beginning and end of each sentence in the input, thereby breaking up encyclopedia articles into manageable segments.

The modules used for preparations of the XML documents, preprocessing, and lexical analysis are written in Python[7].

5.4.3 Text Interpretation and LexTUC

As mentioned above, the text interpretation part of the system is LexTUC, a version of TUC adapted to TUClopedia. The following processing modules are part of LexTUC:

- *Parser,* a natural language parser, described in Section 2.7.1, that uses the Grammar and the Semantic resources to syntactically analyze the tagged word graphs from the Lexical analyzer. The result is a set of parse trees that have already been semantically validated with regard to selectional restrictions.
- *Semantic interpreter,* a module responsible for evaluating the parse trees from Parser and transforming the one that represents the most probable parse into TUC Query Language (TQL), a language based on first-order event logic. To guide this transformation process, information from the other Semantic resources is used. The result is a TQL expression that is passed on to the Reasoning engine.
- *Reasoning engine,* a query processor that receives TQL expressions from the Semantic interpreter and both stores newly acquired knowledge in an application-specific knowledge base (KB) and uses information from the KB and the other Semantic resources in order to answer questions asked by the users of the system.

The Parser, Semantic interpreter, Reasoning engine, and the resources they utilize are all parts of LexTUC.

7 Python, version 2.4.2.

After the natural language parser has extracted the information from an encyclopedia article, the acquired knowledge is stored in the application-specific knowledge base.

LexTUC's Semantic network, including its verb frames, is needed in addition to WordNet, because WordNet does not contain all the information needed during text interpretation, and the verb frames in WordNet are often too general to be useful. However, LexTUC's Semantic network has been modified so that WordNet becomes part of it.

LexTUC, which constitutes the NLU part of the system is written in Prolog[8].

5.5 Multiple-Pattern Approximate Sequence Matching

The implemented sentence-boundary detection, named-entity recognition, and detection of translatable collocations all make use of the same basic algorithms. If one sees the input text as a sequence of words, all problems can be reduced to efficiently matching multiple patterns—possibly with gaps—against the input sequence. That problem corresponds to doing approximate string matching (Navarro 2001) with multiple pattern strings, a problem that is not as well studied as the single pattern case. Therefore, I developed an algorithm for multiple-pattern approximate sequence matching, presented as Algorithm 5.1, with emphasis on both efficiency and legibility that matches multiple patterns in a single run through the input. We also used and presented the same algorithm in (Sætre et al. 2005, 2007).

The algorithm takes four arguments as its input. The first argument is a list, denoted P, that represents the patterns to be matched against. Each element $p \in P$ is a tuple, (T_p, id_p), where id_p denotes the identity—or value—of the pattern (e.g., based on an enumeration of all the patterns), and T_p denotes the pattern—or key—in a tokenized form. Tokenizing a string (possibly) splits it into several shorter strings, based on certain criteria—for example matching of a regular expression (regexp). Through tokenization, the pattern is transformed into a sequence of words. For example, suppose that the patterns

1. `"Great Lake"`,
2. `"Great Lake race"`, and
3. `"Lake Pedder"`

8 SICStus, version 3.11.1.

ALGORITHM 5.1: Identify any pattern $\in P$, occurring in the sequence T, while allowing for gaps designated by a set of ignorable tokens T_{ignore}, by using a carefully constructed hash index I_{P}.

```
1:  function MULTI-PATTERN-MATCH(P, I_P, T, T_ignore)
2:      result ← {}
3:      i ← 0
4:      while i < |T| do
5:          j ← 0                    ▷ Number of matching tokens in T since i.
6:          span ← 0                 ▷ The length of the current match(es) in T.
7:          while (i + span) < |T| do
8:              t ← T[i + span]
9:              if t ∈ T_ignore then
10:                 if span = 0 then
11:                     break        ▷ No path starts with an ignorable token.
12:                 else
13:                     span ← span + 1
14:                     continue
15:             key ← (j, t)
16:             if key ∈ I_P then
17:                 if span = 0 then
18:                     parts ← I_P[key]          ▷ Note that parts is a set.
19:                 else                          ▷ Discard unwarranted paths.
20:                     parts ← I_P[key] ∩ parts
21:                 j ← j + 1
22:                 span ← span + 1
23:                 for all k ∈ {n|(n ∈ parts) ∧ (|P[n][0]| = j)} do
24:                     result ← result ∪ {(i, (i + span), k)}
25:                     parts ← parts − {k}
26:             else
27:                 break
28:         i ← i + 1
29:     return result
```

ALGORITHM 5.2: Build a hash index, I_P, based on the (T_p, id_p) tuples that represent the patterns in P.

```
1: function BUILD-INDEX(P)
2:     I_P ← DICT()
3:     for all T_p, id_p ∈ P do
4:         for all i, t_i ∈ ENUMERATE(T_p) do
5:             if (i, t_i) ∉ I_P then
6:                 I_P[(i, t_i)] ← {}
7:             I_P[(i, t_i)] ← I_P[(i, t_i)] ∪ {id_p}
8:     return I_P
```

represent the patterns we want to match against. Thus, P—with tokenized patterns—becomes

$$\begin{aligned} P = \langle &(\langle \texttt{"Great"}, \texttt{"Lake"} \rangle, id_1), \\ &(\langle \texttt{"Great"}, \texttt{"Lake"}, \texttt{"race"} \rangle, id_2), \\ &(\langle \texttt{"Lake"}, \texttt{"Pedder"} \rangle, id_3) \rangle. \end{aligned} \tag{5.1}$$

The second argument to the algorithm is a hash index, denoted I_P, that is constructed by calling BUILD-INDEX, shown in Algorithm 5.2, with P. For each element $(T_p, id_p) \in P$, several tuples (i, t_i) are generated; where t_i is the ith token in T_p. These tuples are used as the *keys* in I_P, while the *value* associated with each key is a set. Each such set contains the index of each pattern in P that were used to generate the key associated with it. Given the patterns from (5.1), the result of calling BUILD-INDEX(P) becomes a hash index, I_P, that contains the following key → value associations:

$$\begin{aligned} &I_P[(0, \texttt{"Great"})] \rightarrow \{id_1, id_2\}, \\ &I_P[(0, \texttt{"Lake"})] \rightarrow \{id_3\}, \\ &I_P[(1, \texttt{"Lake"})] \rightarrow \{id_1, id_2\}, \\ &I_P[(1, \texttt{"Pedder"})] \rightarrow \{id_3\}, \quad \text{and} \\ &I_P[(2, \texttt{"race"})] \rightarrow \{id_2\}. \end{aligned} \tag{5.2}$$

The third argument of Algorithm 5.1 is a tokenized input string to be matched against, while the fourth argument is a possibly empty set of tokens that the algorithm should ignore if they occur within a matching

region in the input string. Thus, the last argument can be seen as set of stop words.

The returned value from the algorithm is a (possibly empty) set of tuples $\{(n_k, m_k, i_k), \dots\}$, where n_k and m_k are the start and stop indices of the kth match, while i_k is the index of a matching pattern term in P. There will be one such tuple for each matching pattern.

Incorporating the use of a set of ignorable tokens makes the algorithm more robust with regard to noise—for example punctuation—in the input, while preserving the original input. An alternative would be to simply remove the stop words from the input, but that would alter the input data structure so that for example preexisting indices that refer to the original input would become invalid. The non-preserving alternative would also make the returned indices that designate the beginning and the end of a matching region useless without the version of the input that was modified by the algorithm.

Given an ignorable-tokens set $T_{\text{ignore}} = \{\texttt{" "},\texttt{"'s"}\}$, that ignores white space and the possessive *'s*, the patterns P from (5.1), and the index I_P from (5.2), Algorithm 5.1 applied to a tokenized input sequence like

$$T = \langle \texttt{"The"},\texttt{" "},\texttt{"Great"},\texttt{" "},\texttt{"Lake"},\texttt{"'s"},\texttt{" "},\texttt{"race"} \rangle$$

would correctly return the matches designated by the set

$$\{((2,5), id_1), ((2,8), id_2)\}.$$

Presented more graphically, the matches equal

$$\texttt{"the }\underbrace{\texttt{great lake}}_{id_1}\texttt{'s race"},$$

and

$$\texttt{"the }\underbrace{\texttt{great lake's race}}_{id_2}\texttt{"}.$$

It should be noted that the tokenized input preserved the white-space characters, but the algorithm ignored them in adherence to T_{ignore}.

It should also be noted that the pattern identity id_p of multiple patterns can refer to the same value. How this feature can be utilized during named entity recognition will be shown in Section 5.5.2.

5.5.1 Sentence-Boundary Detector

To cut the encyclopedia articles into manageable chunks before parsing, they are split into sentences. However, since the encyclopedia articles

generally contain many abbreviations, determining whether a period ('.') designates the end of a sentence is nontrivial. Therefore, sentence-boundary detector is employed by the lexical analyzer to determine where the boundaries of the sentences are.

The main idea of the sentence-boundary detector implemented in TUClopedia is to recognize the use of abbreviations in the input text. The sentence-boundary detector makes extensive use of the multiple-pattern approximate sequence matching algorithm described above. The patterns to match include 488 abbreviations semi-manually identified in the encyclopedia corpus. The identity id_p, for each pattern T_p, is a tuple containing the form of the abbreviation, and flags that indicate whether the abbreviation may be used to begin or end a sentence. For example, the abbreviation «*m.h.t.*» ("with respect to") can never end a sentence, but «*red.*» ("editor") may.

The corresponding patterns P indexed in I_P are

$$
\begin{aligned}
P = \langle & \ldots, \\
& (\langle \texttt{"m"}, \texttt{"h"}, \texttt{"t"} \rangle, \\
& \quad (\langle \texttt{"med"}, \texttt{"hensyn"}, \texttt{"til"} \rangle, \texttt{may\_begin})), \\
& \ldots, \\
& (\langle \texttt{"red"} \rangle, \\
& \quad (\langle \texttt{"redaktør"} \rangle, \texttt{may\_begin}, \texttt{may\_end})), \\
& \ldots \rangle.
\end{aligned}
$$

The multiple-pattern approximate sequence matching algorithm is then applied with a set of ignorable tokens that, among other things, include periods.

The sentence-boundary detector then basically iterates through each of the period occurrences and decides whether the period is a sentence boundary or not.

5.5.2 Named-Entity Recognition

The named-entity recognizer also makes extensive use of the multiple-pattern approximate sequence matching algorithm presented above.

Titles of works, for example paintings, books, and musical compositions, are matched as simple sequences. Names of persons, however, are matched by permutations automatically generated from the full

form of the name with and without initials. In the case of *John Fitzgerald Kennedy*, the patterns become

$$
\begin{aligned}
P = \langle \ldots, \\
&(\langle \texttt{"J"}, \texttt{"Fitzgerald"}, \texttt{"Kennedy"} \rangle, id_{\texttt{JFK}}), \\
&(\langle \texttt{"J"}, \texttt{"Kennedy"} \rangle, id_{\texttt{JFK}}), \\
&(\langle \texttt{"John"}, \texttt{"F"}, \texttt{"Kennedy"} \rangle, id_{\texttt{JFK}}), \\
&(\langle \texttt{"John"}, \texttt{"Fitzgerald"}, \texttt{"Kennedy"} \rangle, id_{\texttt{JFK}}), \\
&(\langle \texttt{"John"}, \texttt{"Kennedy"} \rangle, id_{\texttt{JFK}}), \\
&(\langle \texttt{"Kennedy"}, \texttt{"J"}, \texttt{"Fitzgerald"} \rangle, id_{\texttt{JFK}}), \\
&(\langle \texttt{"Kennedy"}, \texttt{"J"} \rangle, id_{\texttt{JFK}}), \\
&(\langle \texttt{"Kennedy"}, \texttt{"John"}, \texttt{"F"} \rangle, id_{\texttt{JFK}}), \\
&(\langle \texttt{"Kennedy"}, \texttt{"John"}, \texttt{"Fitzgerald"} \rangle, id_{\texttt{JFK}}), \\
&(\langle \texttt{"Kennedy"}, \texttt{"John"} \rangle, id_{\texttt{JFK}}), \\
&(\langle \texttt{"Kennedy"} \rangle, id_{\texttt{JFK}}), \\
&\ldots \rangle,
\end{aligned}
$$

Lexical Analyzer (Multitagger)

where $id_{\texttt{JFK}}$ designates a system-internal reference to automatically extracted biographic information about Kennedy.

5.6 Lexical Analyzer (Multitagger)

The multitagger delegates tasks both to the sentence-boundary detector and to the named-entity recognizer. When results are returned from these submodules, the multitagger is responsible for detecting any inconsistencies and cleaning up the word graph.

For example, a biography article in *Store norske leksikon* about the author *Jo Nesbø* contains an interesting passage where a title tag crosses a period ('.'), as shown from the XML tags in the original source:

> ... gitt ut <vtit type="LITT">Stemmer fra Balkan. Atten dager i mai</vtit> (1999) sammen ...

The sentence-boundary detector tags the period as the end of a sentence, causing the erroneously analysis

$$\underbrace{\ldots \text{gitt ut Stemmer fra Balkan.}}_{\text{sentence}_1} \quad \underbrace{\text{Atten dager i mai (1999) sammen} \ldots .}_{\text{sentence}_2}$$

However, the named-entity recognizer correctly recognizes the title, as in

... gitt ut Stemmer fra Balkan. Atten dager i mai (1999) sammen ....
litteratur$_1$

By preferring the title-of-work tag over end-of-sentence tags, the multitagger correctly avoids errors such as the one introduced by the sentence-boundary detector.

It should be noted that during normal use of the system, the XML tags are ignored during preprocessing. The semistructured information is only used when building the onomasticon.

5.7 Results

This section presents results from the implemented TUClopedia system. Hence, it also shows how Verto is applied in TUClopedia. It should be noted that these are only some preliminary results, because the implementation of TUClopedia is not finished yet.

Most of the modules in the system are already complete. However, the integration of the modules from LexTUC with the rest of the system has just begun. This means that the current version of the system can retrieve all the encyclopedia articles from the database; create a large onomasticon based on the semistructured articles; perform lexical analysis of all input, both articles and questions; parse, semantically analyze, and acquire knowledge from a few arbitrary selected articles; and answer questions related to the acquired knowledge.

5.7.1 *First Example*

The first example shows how knowledge is acquired from parsing a very short[9] article from *Store norske leksikon*. The article is about the A+ programming language (Henriksen 2003):

> A+
>
> programmeringsspråk for datamaskiner, avledet av APL.
>
> [A+
>
> programming language for computers, derived from APL.
>
> (My translation.)]

9 Actually, 23 of the first 100 articles of the encyclopedia are shorter than or have the same length as the one used in this example.

```
txt(w('A+', name('A+', n, programming_language_n_1)), 0, 1).
txt(w('er', verb(be_v_2, pres, fin)), 1, 2).
txt(w('programmeringsspråk',
    noun(programming_language_n_1, plu, u, n)), 2, 3).
txt(w('programmeringsspråk',
    noun(programming_language_n_1, sin, u, n)), 2, 3).
txt(w('for', prep('for')), 3, 4).
txt(w('datamaskiner', noun(computer_n_1, plu, u, n)), 4, 5).
txt(w('avledet', verb(derive_v_3, past, part)), 5, 6).
txt(w('avledet', ['avledet']), 5, 6).
txt(w('av', prep('from')), 6, 7).
txt(w('av', prep('of')), 6, 7).
txt(w('av', prep('off')), 6, 7).
txt(w('APL', name(apl, n, programming_language_n_1)), 7, 8).
txt(w('.', ['.']), 8, 9).
```

Listing 5.1: Word graph for the A+ article.

Lexical Analysis

The article gets represented by the word graph as shown in Listing 5.1. The lexical analyzer currently ignores commas (','). The first line of text in an encyclopedia article does not constitute a full sentence. However, the lexical analyzer automatically turns it into a sentence by joining it together with the article heading and the added verb be_v_2[10].

It should be noted that the word graph had to be manually edited in three ways. These edits will be discussed below.

First of all, the verb, *avledet*, was originally mapped in Ordnett to the senses shown in Table 5.1 on the next page. While some of the senses definitely hint at the meaning of the verb in the Norwegian article, none of them really covers the intended meaning. The Norwegian verb *avlede* mainly has two senses, to "divert", "deflect", or "redirect" and to "derive". However, only one of those meanings is covered by Ordnett. Therefore, those word-graph entries were manually replaced with derive_v_3[11], which represents the *other* meaning of «*avlede*».

10 WordNet gloss: be identical to; be someone or something; "The president of the company is John Smith"; "This is my house."

11 WordNet gloss: come from; "The present name derives from an older form".

TABLE 5.1: Glosses for the target WordNet senses of the mappings from the Norwegian verb *«avledet»*.

Sense	Gloss
`deflect_v_1`	prevent the occurrence of; prevent from happening; "Let's avoid a confrontation"; "head off a confrontation"; "avert a strike"
`deflect_v_2`	turn from a straight course, fixed direction, or line of interest
`deflect_v_3`	turn aside and away from an initial or intended course
`deflect_v_4`	draw someone's attention away from something; "The thief distracted the bystanders"; "He deflected his competitors"
`divert_v_1`	turn aside; turn away from
`redirect_v_1`	channel into a new direction; "redirect your attention to the danger from the fundamentalists"

Thus, one can argue that this manual interference could have been avoided if Ordnett's coverage was higher.

In one respect, LexTUC does not care about symbol names, it will only use the symbol names for computation. Therefore, one could be tempted to argue that it does not matter what particular symbols the individual constituents of a text or sentence is translated into, as long as the same symbols will be chosen every time that the same words are encountered. For example, if the system consequently maps the concept "jet plane" to "cat", acquired knowledge about jet planes will internally be linked to "cat". Therefore, in such a situation, given a question like "What is the typical speed of a jet plane?", the question could be answered correctly—even though, internally the system retrieves information about a "cat" traveling at 850 km/h.

However, there are at least two main reasons that such unfortunate mappings should be manually overridden. First, what might seem like an inaccuracy at one level, will be completely wrong at another. Consider for example what would happen if the reasoning engine needed

$$
\begin{aligned}
\exists x_1(((&\text{isa}(\texttt{programming\_language\_n\_1}, \texttt{real}, x_1) \\
&\wedge (\exists x_2((\text{isa}(\texttt{computer\_n\_1}, \texttt{real}, x_2) \\
&\quad \wedge \exists x_3((\text{event}(x_3) \wedge (\text{agent}(x_1, x_3) \\
&\quad\quad \wedge ((\text{present}(\texttt{real}, x_3) \wedge \text{action}(\texttt{be\_v\_4}, x_3)) \\
&\quad\quad\quad \wedge \text{mod}(\texttt{for}, x_2, x_3))))))) \\
&\wedge \exists x_4((\text{event}(x_4) \wedge ((\text{present}(\texttt{real}, x_4) \wedge \text{action}(\texttt{derive\_v\_3}, x_4)) \\
&\quad\quad \wedge (\text{patient}(x_1, x_4) \\
&\quad\quad \wedge (\text{agent}(\texttt{pro}, x_4) \wedge \text{mod}(\texttt{from}, \texttt{apl}, x_4)))))))) \\
&\wedge \exists x_5((\text{event}(x_5) \wedge (\text{agent}(\texttt{aplus}, x_5) \wedge ((\text{present}(\texttt{real}, x_5) \\
&\quad\quad\quad\quad \wedge \text{action}(\texttt{be\_v\_2}, x_5)) \\
&\quad\quad\quad\quad \wedge \text{patient}(x_1, x_5)))))))
\end{aligned}
$$

FIGURE 5.2: Logical formula representing the contents of the A+ article.

to use hypernymy or synonymy rules to complete an inference. The inference would be wrong because the incorrect mapping's semantics is implicitly defined by its relations to other concepts in the predefined ontology. Second, allowing such incorrect mappings would entail them spreading throughout the system, for example to manually defined semantic rules used during parsing and reasoning about queries. Maintaining such a system would quickly become an unmanageable task.

The second manual edit of the word graph in Listing 5.1 on page 164 was to change the tags of A+ and APL from `entity_n_1` to `programming_language_n_1`. The reason for this was that there were no entries in the onomasticon covering those named entities at the time of tagging. However, such information about all the keywords in the encyclopedia could quite easily be extracted using a shallow parser that exploits the encyclopedia's semistructured information.

The third manual edit of the word graph in Listing 5.1 was to remove superfluous entries that were removed during semantic disambiguation anyway.

The Parser correctly disambiguates the input and generates a parse tree that, through application of Montagovian semantics (see, for example, Gamut 1991a; pp. 139–221), is transformed into the rather elaborate first-order predicate logic expression shown in Figure 5.2 on the preceding page. The Semantic interpreter then transforms the logical expression into a Skolemized TQL expression that represents the acquired knowledge, as shown in Listing 5.2 on the following page. The **sk**(*n*) terms represent Skolem-constants, and every event—for example **sk**(3), **sk**(4), and **sk**(5) in Listing 5.2—functions as a link between the semantic units and verbal complements that make up the newly acquired knowledge.

As can be seen from the acquired knowledge, the parser has correctly chosen the correct interpretation of a syntactically ambiguous sentence, even though the comma in the original text was ignored. LexTUC correctly interpreted the sentence as a situation where A+ is a programming language to be used with computers, and that the programming language was derived from APL by *somebody* (designated by the "pro" agent). Two erroneous interpretations were avoided. The first is where A+ is for computers that have been derived from APL. The second is where A+ is derived by APL (because the Norwegian «*av*» can be translated as both "from", "by", and "off").

5.7.2 *Second Example*

With minor changes to some of the word graph constituents—similar to the ones mentioned in Section 5.7.1—the system was able to interpret the following article about the murder of *John Fitzgerald Kennedy* (Henriksen 2003):

> Drapet på Kennedy.
>
> Presidenten ble myrdet av en snikskytter 22. november 1963 under et besøk i Dallas, Texas. Kennedy-mordet ble umiddelbart gjenstand for en rekke ulike teorier og påstander som har fortsatt å oppta allmennheten etterpå. Avgjørende vekt ble tillagt den bredt oppnevnte offisielle granskingskommisjonen ledet av høyesterettsjustitiarius Earl Warren. Warren-kommisjonens rapport fra 1964 la mordansvaret på Lee Harvey Oswald (1939–63) alene. Oswald ble arrestert rett etter drapet, men nektet seg skyldig. Han ble 24. november 1963 skutt ned og drept for åpent kamera av nattklubbeieren Jack Ruby (1911–67) på Dallas' hovedpolitistasjon.

Results

```
% ''A+ programmeringsspråk for datamaskiner, avledet av APL.''
% (A+ programming language for computers, derived from APL.)

isa(programming_language_n_1, real, sk(1))
isa(computer_n_1, real, sk(2))

% Programming language, sk(1), is for computer, sk(2).
event(sk(3))
agent(sk(1), sk(3))
present(real, sk(3))
action(be_v_4, sk(3))
mod(for, sk(2), sk(3))

% Somebody (pro) derives programming language, sk(1), from 'APL'
event(sk(4))
present(real, sk(4))
action(derive_v_3, sk(4))
patient(sk(1), sk(4))
agent(pro, sk(4))
mod(from, 'APL', sk(4))

% A+ is programming language, sk(1).
event(sk(5))
agent('A+', sk(5))
present(real, sk(5))
action(be_v_2, sk(5))
patient(sk(1), sk(5))
```

LISTING 5.2: KB representation of the knowledge acquired from the A+ article, expressed through TQL expressions. The situation each event describes is paraphrased in the comments, following the '%' signs. The original text is included as a comment at the beginning of the listing.

Dødsdommen over Ruby (1964) ble ikke fullbyrdet idet Ruby døde av sykdom kort etterpå. Warren-kommisjonens rapport har siden blitt kritisert, men det har ikke kommet frem opplysninger som har kunnet verifisere en annen versjon. Kennedy-mordet har derfor fortsatt å være en gåte.

Kennedy ble gravlagt på æreskirkegården Arlington utenfor hovedstaden Washington D.C. I årene etter sin død ble Kennedy regnet som en av de mest betydningsfulle presidenter i USA. Hans ungdommelige stil, hans veltalenhet og sjarm innebar en fornyelse av amerikansk politikk og samfunnsliv. Han og hans kone Jacqueline (se J. Onassis) var samtidens mest berømte par. Senere har bildet av ham blitt noe mer nyansert; det er bl.a. pekt på manglende resultater i innenrikspolitikken, hans anstrengte forhold til Kongressen og at hans privatliv muligens var noe mer frynsete enn samtiden fikk inntrykk av.

[The Kennedy Assassination.

The President was assassinated November 22, 1963 during a visit to Dallas, Texas. The Kennedy assassination immediately became subject to a range of different theories and allegations that subsequently have continued to occupy the general public. Great significance was attached to the widely appointed public commission lead by Chief Justice of the Supreme Court Earl Warren. The Warren Commission's report from 1964 placed the responsibility for the murder on Lee Harvey Oswald (1939–63) alone. Oswald was arrested immediately after the murder, but pleaded not guilty. On November 24, 1963 he was shot down and killed before live cameras by the nightclub owner Jack Ruby (1911–67) at the Dallas Police Headquarters. Ruby's death sentence (1964) was not executed because Ruby died from disease shortly thereafter. The Warren Commission's report has since been criticized, but no information that has been able to verify a different version has emerged. The Kennedy assassination has therefore continued to be an enigma.

Kennedy was buried at the Arlington memorial cemetery outside the capital Washington D.C. In the years after his death, Kennedy was recognized as one of the most important presidents of the USA. His youthful style, his eloquence and charm involved a renewal of American politics and community. He and his wife, Jacqueline (see J. Onassis)

were the most famous couple of their era. His image has since become a little more nuanced; it has, among other things, been pointed to a lack of results in domestic policy, his tense relationship with Congress, and that his private life possibly was a little more dubious than the impression his contemporaries gained. (My translation.)]

Results

The first sentence was interpreted by LexTUC into the TQL expressions shown in Listing 5.3 on the next page. The TQL expressions were added to the TUClopedia's KB as newly acquired knowledge. The interpretation seems very reasonable.

After having processed all the other sentences of the article in a similar manner, TUClopedia could answer questions about the text.

Question Answering

For example, given the question *«Hvem myrdet presidenten?»* ("Who murdered the president?"), the Semantic interpreter transforms the question into the TQL query expression shown in Listing 5.4 on page 172.

The answer to the question is deduced from the TQL query expression that represents the question and the knowledge already acquired by the system.

The proof performed by LexTUC to answer the TQL query from Listing 5.4 is shown in Listing 5.5 on page 172. The comments in the listing show how the variables from the TUC Query Language query (TQL query) in Listing 5.4 are unified with the facts from the KB that were shown in Listing 5.3. Interestingly, one can see how LexTUC applied a rule to search WordNet for specializations of `somebody_n_1`. Hence, LexTUC could use the fact that, indirectly, a `sniper_n_1` is a kind of `somebody_n_1`, to satisfy the query constraint **isa**(`somebody_n_1`, real, A) with the fact **isa**(`sniper_n_1`, real, **sk**(2)) that was present in the KB.

As can be seen, the answer **sk**(2) translates to "a sniper", which is correct given the information in the KB that was shown in Listing 5.3.

It is outside the scope of TUClopedia to generate natural language answers. However, the current implementation gives an insight into some of the considerations that should be taken if such answers were to be generated. Certain of these considerations will be discussed below.

Complex Entailment

To deduce an answer to the previous question was rather straightforward. The question *«Når døde presidenten?»* ("When did the president

```
% «Presidenten ble myrdet av en snikskytter 22. november 1963 under
% et besøk i Dallas, Texas.» (''The President was assassinated November
% 22, 1963 during a visit to Dallas, Texas.'')

isa(president_n_5, real, sk(3))
isa(dallas_n_1, real, sk(4))

% There is a visit, sk(3), in Dallas, sk(4).
event(sk(5))
agent(sk(3), sk(5))
present(real, sk(5))
action(be_v_4, sk(5))
mod(in, sk(4), sk(5))

isa(texas_n_1, real, sk(6))

% The visit, sk(3), is in Texas, sk(6).
event(sk(7))
agent(sk(3), sk(7))
present(real, sk(7))
action(be_v_4, sk(7))
mod(in, sk(6), sk(7))

isa(president_n_3, real, sk(1))
isa(sniper_n_1, real, sk(2))

% The President, sk(1), was murdered by a sniper, sk(2) during the
% visit, sk(3), on November 22, 1963.
event(sk(8))
past(real, sk(8))
action(murder_v_1, sk(8))
patient(sk(1), sk(8))
agent(sk(2), sk(8))
mod(under, sk(3), sk(8))
mod(nil, date(1963, 11, 22), sk(8))
```

LISTING 5.3: KB representation of the knowledge acquired from the first sentence in the article about the assassination of Kennedy. The original sentence is included as a comment at the beginning of the listing. Each event is paraphrased in the comments.

Results

```
% «Hvem myrdet presidenten?» (''Who murdered the president?'')

which(A),
isa(somebody_n_1, real, A),
isa(president_n_3, real, B),
agent(A, C),
action(murder_v_1, C),
past(real, C),
event(C),
patient(B, C)
```

LISTING 5.4: TQL query representation produced by LexTUC from a question about who murdered Kennedy. The original question is included as a comment at the beginning of the listing. The TQL query can be paraphrased as "Which A was the agent in a murder event, C, where a president, B, was the patient?"

```
% Proof for answering the question «Hvem myrdet presidenten?» (''Who
% murdered the president?'')

which(sk(2))
isa(somebody_n_1, real, sk(2)) <=
   isa(person_n_1, real, sk(2)) <=
      isa(expert_n_1, real, sk(2)) <=
         isa(sniper_n_1, real, sk(2))          % A = sk(2).
isa(president_n_3, real, sk(1))                % B = sk(1).
agent(sk(2), sk(8))                                % C = sk(8).
action(murder_v_1, sk(8))
past(real, sk(8))
event(sk(8))
patient(sk(1), sk(8))

=>

sk(2)
```

LISTING 5.5: The proof performed by LexTUC to answer the TQL query shown in Listing 5.4. The comments show how the variables from the query are unified with facts from the KB.

```
% «Når døde presidenten?» (''When did the president die?'')

which(A),
isa(president_n_3, real, B),
agent(B, C),
action(die_v_1, C),
past(real, C),
event(C),
mod(in, A, C),
isa(time_n_1, real, A)
```

LISTING 5.6: TQL query representation of a question about when Kennedy died. The TQL query can be paraphrased as "Which A is the time of an event, C, in which the President, B, died?"

die?") looks just as simple as the previous question. However, as becomes clear when studying the corresponding TQL expression in Listing 5.6, some additional information is needed to provide an answer. The reason is that the question uses another verb than *murder*. There is no explicit knowledge about any *die* action.

To remedy this situation, a complex entailment rule, shown in Listing 5.7 on the next page, was added to LexTUC. The rule can be paraphrased as "if X the patient of a murder in an event E1, then there also is a coinciding event E2 where X is an agent who dies." The rule also ensures that the same temporal modifiers will be applied to both events.

As the proof in Listing 5.8 on page 175 shows, by applying the rule that being murdered entails dying, TUClopedia is able to deduce the correct answer to the question; that is, November 22, 1963.

Hypernymy

The final QA example relates to the knowledge acquired regarding where Kennedy was buried. The relevant acquired knowledge is shown in Listing 5.9 on page 176.

Given the question «*Hvor ble Kennedy gravlagt?*» ("Where was Kennedy entombed?"), LexTUC generates the TQL expression in Listing 5.10 on page 176.

```
coincide(E1, sk(die, X, E1)) <==
  (event(E1),  action(murder_v_1, E1),  patient(X, E1)).
action(die_v_1, sk(die, _X, _E1)) <== true.
agent(X, sk(die, X, _E1)) <== true.

mod(in, D, sk(die, _, E1)) <== mod(in, D, E1).

event(F) <==
  (event(E),
  coincide(E, F)).

present(Real, F) <==
  (event(E),
  present(Real, E),
  coincide(E, F)).

past(Real, F) <==
  (event(E),
  past(Real, E),
  coincide(E, F)).

future(Real, F) <==
  (event(E),
  future(Real, E),
  coincide(E, F)).
```

LISTING 5.7: Temporal reasoning rules that state that "you die if someone murders you."

```
% Proof for answering the question «Når døde presidenten?» (''When did
% the president die?'')

which(date(1963, 11, 22))
isa(president_n_3, real, sk(1))                    % B = sk(1).
agent(sk(1), sk(die, sk(1), sk(8)))                  % C = sk(die,sk(1),sk(8)).
action(die_v_1, sk(die, sk(1), sk(8)))
past(real, sk(die, sk(1), sk(8))) <=
   event(sk(8))
   past(real, sk(8))
   coincide(sk(8), sk(die, sk(1), sk(8))) <=
      event(sk(8))
      action(murder_v_1, sk(8))
      patient(sk(1), sk(8))
event(sk(die, sk(1), sk(8))) <=
   event(sk(8))
   coincide(sk(8), sk(die, sk(1), sk(8))) <=
      event(sk(8))
      action(murder_v_1, sk(8))
      patient(sk(1), sk(8))
mod(in, date(1963, 11, 22), sk(die, sk(1), sk(8))) <=
   mod(in, date(1963, 11, 22), sk(8)) <=              % A = date(1963, 11, 22).
      mod(nil, date(1963, 11, 22), sk(8))
isa(time_n_1, real, date(1963, 11, 22))

=>

date(1963, 11, 22)
```

LISTING 5.8: The proof performed by LexTUC to answer the TQL query shown in Listing 5.6. The comments show how the variables from the query are unified with facts from the KB shown in Listing 5.3.

```
% «Kennedy ble gravlagt på æreskirkegården Arlington utenfor
% hovedstaden Washington D.C.» (''Kennedy was buried at the Arlington
% memorial cemetery outside the capital Washington D.C.'')
```

Results

```
isa(arlington_n_1, real, sk(47))
isa(churchyard_n_1, real, sk(48))

% JFK was entombed at a churchyard, sk(48), outside Arlington, sk(47).
event(sk(49))
past(real, sk(49))
action(entomb_v_1, sk(49))
patient(john_fitzgerald_kennedy_p_86918, sk(49))
agent(pro, sk(49))
mod(outside, sk(47), sk(49))
mod(at, sk(48), sk(49))
```

LISTING 5.9: The relevant knowledge acquired by LexTUC about where Kennedy was entombed. The original sentence is included in the comment at the beginning of the listing.

```
% «Hvor ble Kennedy gravlagt?» (''Where was Kennedy entombed?'')

which(A),
action(entomb_v_1, B),
past(real, B),
patient(john_fitzgerald_kennedy_p_86918, B),
agent(pro, B),
mod(at, A, B),
isa(location_n_1, real, A),
event(B)
```

LISTING 5.10: TQL query expression representing a question about where Kennedy was entombed. The query may be paraphrased as "Which A is a location where Kennedy was entombed in an event, B?"

```
% Proof for answering the question «Hvor ble Kennedy gravlagt?»
% (''Where was Kennedy entombed?'')

which(sk(48))
action(entomb_v_1, sk(49))
past(real, sk(49))
patient(john_fitzgerald_kennedy_p_86918, sk(49))
agent(pro, sk(49))
mod(at, sk(48), sk(49))
isa(location_n_1, real, sk(48)) <=
   isa(region_n_3, real, sk(48)) <=
      isa(geographical_area_n_1, real, sk(48)) <=
         isa(churchyard_n_1, real, sk(48))
event(sk(49))

=>

sk(48)
```

LISTING 5.11: The proof performed by LexTUC to answer the TQL query shown in Listing 5.10.

The TQL query can be paraphrased as "which A is a `location_n_1` where Kennedy was entombed in an event, B?", which quite correctly asks for a location. However, in the acquired knowledge, the place that Kennedy was entombed at is a `churchyard_n_1`, not at `location_n_1`.

Just as in the example in Section 5.7.2, because LexTUC's Reasoning engine has access to semantic relations defined in WordNet and rules to explore those relations, LexTUC is able to infer a correct answer, as shown in Listing 5.11. The proof shows how LexTUC is able to use the fact that a churchyard (indirectly) is a kind of location to provide the correct answer "at a churchyard".

WordNet also contains hypernymy relations for verbs. For example, according to WordNet, `murder_v_1` is a hyponym of `kill_v_1`. Therefore, LexTUC can use this information to correctly answer a question like *«Hvem drepte Kennedy?»* ("Who killed Kennedy?").

Users' Expectations

The example in the previous section showed that even though LexTUC is able to answer questions correctly and *very* concisely, the answer is not necessarily what a user would expect from a helpful QA system. If a human asks where a famous person is buried, the answer "at the churchyard" is probably not among the anticipated answers. The answer is clearly underspecified. For example, a more valuable answer—for which the system already has acquired the necessary knowledge—would be "at a churchyard outside Arlington."

As mentioned above, Natural Language Generation (NLG) is outside the scope of TUClopedia. Nonetheless, if a more polished user interface should be added to the system, a NLG module could be part of such a solution. However, if polishing the user interface means hiding the core details, like the proofs performed to provide answers, users may question the correctness of the answers; or, indirectly question the credibility of the system. On the other hand, users unable to comprehend the proofs would probably not be comforted by their presence either.

One way to increase the probability that users perceive the system as credible is through transparency. By providing the users with text snippets that contain the source of the acquired knowledge, the users can easily check that the answer is in accordance with the sources.

5.8 Analysis and Discussion

With regard to the questions by Paşca (2003), quoted in Section 5.1, TUC's natural language analysis and TQL knowledge representation seem well suited to the task of correctly interpreting questions and finding the correct answer. However, some observations indicate that caution must be exercised to ensure the scalability of the TUClopedia system.

5.8.1 The Scalability of TUClopedia

The current results from the TUClopedia system are not enough to decisively tell how well the currently implemented approach scales with regard to both the input size and to the growing size of the KB. However, the preliminary results and experiences give a few indications of challenges that must be overcome to make the system ready for full-scale real-world usage. Next, I will present some observations and assumptions related to the scalability issues involved and some ideas on how the system's scalability can be improved.

Findings and Assumptions

The *Store norske leksikon* (Henriksen 2003) encyclopedia contains approximately 651,059 sentences.[12] In the examples in Sections 5.7.1 and 5.7.2 the average number of events defined through the extracted knowledge is 2.33. Furthermore, each such event is on the average related to 7.86 Prolog facts through predicates like `isa/3`, `patient/2`, and `mod/3`. To simplify this discussion we can assume that most of these Prolog facts are unique per event.

Assuming that these numbers are representative for the rest of the other encyclopedia articles, we can get a rough idea of the sizes involved. Simply multiplying these numbers with the number of sentences yields $1,516,967 \approx 1.5 \cdot 10^6$ events that will be related to $11,923,364 \approx 12 \cdot 10^6$ Prolog facts. Even though these numbers are based on very sparse information, and the facts related to each event might be related to other events too (thus reducing the number of *unique* Prolog facts), they might give a vague idea about the potential challenges that must be conquered for this approach to be viable.

In the original plan, TUClopedia was supposed to first parse the complete encyclopedia in one big batch job to acquire knowledge, and then use the acquired knowledge to answer questions from users.

Currently, TUClopedia on the average uses 293.33 ms of central processing unit (CPU) time per sentence it interprets, running under a single SICStus version 3.12.7 process on version 2.6.27 of the Linux kernel on a computer with an Intel Core Duo T2400 dual core 1833 MHz CPU with a 64 KB level 1 cache and a 2 MB level 2 cache and 2 GB of random access memory (RAM).

Given that all of the sentences were to be processed by a single TUClopedia process, the system would use approximately 53 hours, or 2.21 days and nights, to process them. If, however, the average time spent on processing each sentence empirically should prove to be, for example, 1 s then the whole process would take about 181 hours, or 7.5 days and nights.

Hence, we can assume that the whole initial interpretation of the encyclopedia will take between 2.21 and 7.5 days and nights using this rather naïve approach. The length of the initialization period does not necessarily pose a problem, if it is a one-time cost. However, whenever any of the components involved changes substantially, one might want to repeat the complete initial interpretation step. This will probably be the case both during further development of the system and in a

12 This number is based on the output from my algorithm for splitting paragraphs into sentences, described in Section 5.5.1.

production setting. Thus, reducing the time TUClopedia spends on interpreting the complete encyclopedia may prove important.

Another challenge is that the encyclopedia contains 128,822 different Norwegian words, including different inflections, but not words found in tagged titles and names. Of the Norwegian words that occur in the encyclopedia, 55,772 can directly be found in NorKompLeks. For the TUClopedia approach to scale appropriately, it is of paramount importance that the system is actually able to recognize the words used in input text.

Possible Solutions

To comment on the last of the above observations and assumptions first, the vast majority of the 73,050 words not found in NorKompLeks are compounds. Those words can therefore be automatically broken down into their constituents and analyzed by the compound-word analyzer. Consequently, they should not represent a real problem to TUClopedia.

The words that neither occur directly in NorKompLeks nor can be automatically analyzed are proper names and certain uncommon words; the reason they are present in the encyclopedia. These words must be added manually.

The findings and assumptions described above also indicate that handling the sheer size of the input, and therefore also the size of the KB, must be given some consideration.

One way to reduce the total time used interpreting the encyclopedia articles would be to parallelize the interpretation process. The process of mass-interpreting encyclopedia articles is well suited for parallelization because the interpretation of each article can easily be delegated to one of multiple processes running in parallel. The parallel processes may run on separate computers, or on separate CPU cores, but must share the same KB.

Parallelization often comes with an overhead caused by an increased need for interprocess synchronization and data communication (Patterson and Hennessy 1998). However, in a networked cluster, each host, or node, can keep its own copy of all read-only data, while all the nodes read from and write to the shared KB that can be hosted on a central master node. If necessary, even the KB could be duplicated as slaves on each node for read-only access, while updates to the KB would be written to the master KB. Each update to the KB would then need to be distributed to all the slave nodes, but in total the parallelization overhead should be negligible.

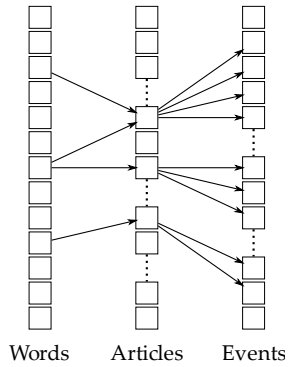


FIGURE 5.3: Example of an index that maps words to articles they occur in and an index that map those articles to knowledge extracted from them. The example is grossly simplified.

Furthermore, building a networked parallel cluster from commodity hardware is not necessarily very costly (Constantinescu-Fulöp and Cassens 2003).

By applying, for example, ten clustered computers to the task of interpreting encyclopedia articles in a distributed, parallel fashion the wall-clock time needed to complete the task is reduced to between 5.2 and 18.1 hours.

The third concern the above observations and assumptions raise is related to the growing size of the KB as the complete encyclopedia gets parsed. It seems unlikely that TUClopedia will be able to answer questions about its acquired knowledge with satisfactory response times if the system will have to reason about the question with *all* the events and facts in the KB available at the same time.

To improve the response time of the QA system, a possible solution is to dynamically load only a smaller, more relevant subset of the complete KB into memory, depending on the received input. That way, TUClopedia will avoid having to consider large amounts of irrelevant knowledge when it performs proofs to answer questions. For TUClopedia to extract subsets of its complete KB, relevant to the input text, one could index acquired knowledge by establishing mappings both from the identifier of the article that the knowledge was acquired from and from each word in the article to the article identifier, as shown in Figure 5.3.

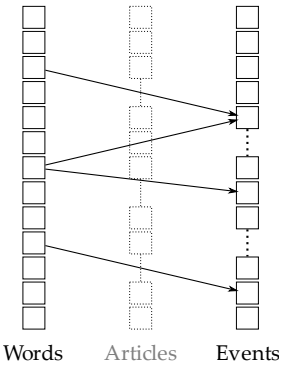


FIGURE 5.4: Example of an index that maps words to knowledge that was extracted form sentences they occurred in. Compared to Figure 5.3, this scheme omits the use of the article-to-events index and will therefore likely produce KB subsets with fewer events and facts per input word.

Given the existence of such indices, TUClopedia could use them to retrieve a subset of the KB by following the mappings from each word that occur in the input, via the articles that those words have occurred in earlier, to facts and events that therefore probably are relevant.

This attempt defines facts' and events' *relevancy* to input words based on whether the same words were used in articles from which the knowledge stems from. The events and facts retrieved by using these indices are probably relevant because they originate from articles where the same words were used, and it seems safe to assume that each article comprises a coherent context. This indexing scheme also takes into account the ambiguity of words because a particular word may have been interpreted differently, depending on the context that each sentence and article forms, thereby establishing mappings from that word to acquired knowledge where different senses of the word occur.

Another solution would be to simply create an index that maps the article words to the set of knowledge extracted from each sentence that the words occurred in, as shown in Figure 5.4. However, by using an indirect indexing like first described, TUClopedia should be able to gather a higher number of relevant events and facts, because most articles consist of more than one sentence.

No matter which one of the indexing schemes for creating dynamic subsets of the KB are used, care must be taken so that its use does not degrade the precision and recall of the QA system.

5.8.2 Ordnett's Suitability

The preliminary results from the implementation of TUClopedia are in accordance with Verto's high precision and lower recall reported in Section 4.11.[13]

If a mapping was not found in Ordnett, the Lexical analyzer tried to find one by consulting the Translation framework. However, some of the mappings not found in the Ordnett resource had to added manually. These were mostly simplex words.

On the other hand, the Translation framework succeeded several times when confronted with Norwegian compounds as input. For the time being, when confronted with compound mappings, LexTUC applied a simple strategy to simply use only their heads in its analyses.

5.8.3 WordNet's Lacking Verb Frames

Others (Kipper et al. 2000a, 2004; Kwon and Hovy 2006) have criticized WordNet for its lacking verb frames, shown in Table 5.2.

We share the experience that the verb frames are too general to be used for selectional restrictions. Therefore, we had to manually add more specific verb frames to LexTUC.

Making such manual additions for all the relevant verbs in an open domain corpora sums up to become a huge task to complete. Therefore, a possible future direction for TUClopedia would be to incorporate a semantic resource like VerbNet (Kipper et al. 2000a, 2004) or FrameNet (Fillmore et al. 2003; Baker et al. 2003) that provide much more detailed information about selectional restrictions and the semantics of verbs.

Those resources are compatible with WordNet. Therefore, such an approach would also benefit from the mappings that Ordnett provides.

5.8.4 Providing Answers to Queries About Encyclopedia Content

There are several possible responses that can be considered adequate for a question regarding the contents of encyclopedia articles. For example, a system could simply return references to the articles that match the

13 Verto is the mapping framework developed in Chapter 4.

TABLE 5.2: WordNet's generic sentence frames for verbs (Source: Kohl et al. 1998).

Frame	Generic Sentence
1	Something —s
2	Somebody —s
3	It is —ing
4	Something is —ing PP
5	Something —s something Adjective/Noun
6	Something —s Adjective/Noun
7	Somebody —s Adjective
8	Somebody —s something
9	Somebody —s somebody
10	Something —s somebody
11	Something —s something
12	Something —s to somebody
13	Somebody —s on something
14	Somebody —s somebody something
15	Somebody —s something to somebody
16	Somebody —s something from somebody
17	Somebody —s somebody with something
18	Somebody —s somebody of something
19	Somebody —s something on somebody
20	Somebody —s somebody PP
21	Somebody —s something PP
22	Somebody —s PP
23	Somebody's (body part) —s
24	Somebody —s somebody to INFINITIVE
25	Somebody —s somebody INFINITIVE
26	Somebody —s that CLAUSE
27	Somebody —s to somebody
28	Somebody —s to INFINITIVE
29	Somebody —s whether INFINITIVE
30	Somebody —s somebody into V-ing something
31	Somebody —s something with something
32	Somebody —s INFINITIVE
33	Somebody —s VERB-ing
34	It —s that CLAUSE
35	Something —s INFINITIVE

topic of the question. This would constitute document retrieval behavior, well known from today's common Web search engines. Such behavior could be implemented using shallow indexing methods, like some variation of the vector-space model (VSM) (Salton et al. 1975) or manual phrase indexing. Phrase, or keyword, searches are already offered by on-line encyclopedias.[14] However, these systems do not interpret the questions semantically, and hence cannot provide answers based on deeper semantics of the question.

An improvement over phrase searches would be to interpret the text of the encyclopedia in order to extract the semantic knowledge, like TUClopedia does. Questions could then be answered either by providing references to the articles that must be read in order to answer the question, or by providing a concise answer to the question, or a combination of these answer forms.

14 Examples of on-line encyclopedias that offer phrase searches include Store Norske Leksikon (`http://www.storenorskeleksikon.no/`) and Encyclopædia Britannica (`http://www.britannica.com/`)

Analysis and Discussion

6

This chapter will provide some further analysis and discussion of the results presented in chapters 3, 4, and 5, focusing on a few topics that are not easily confined in any single of those chapters. For example, it will show how the semantic information in Ordnett can be used to improve the automatic analysis of Norwegian compounds. Finally, it will provide the answers to the research questions from Section 1.3, and summarize the contribution of the research presented herein.

6.1 Automatic Analysis of Compounds Based on Semantics

The Ordnett resource may be used for several applications; one application could be to improve the automatic analysis of Norwegian compound words.

Several schemes for interpreting compound nouns have been presented in the literature. For example, Lauer (1995) proposed a way to classify compound nominals by paraphrasing them using the eight prepositions "of", "for", "in", "about", "with", "from", "on", and "at". For example, "a baby chair" means "a chair *for* babies".

Furthermore, discussing how to interpret and classify compound nouns, Copestake and Briscoe (2005) propose a classification scheme where a compound is said to belong to the first of the categories presented in Table 6.1 that fits (in the order they are presented). The *deverbal* category refers to nouns where the head of the compound is a deverbal nominal. A relational noun is a noun that describes a relation between two entities, such as "sport supporter"

$$\text{sport}(x) \wedge \text{supporter}(y, x) \tag{6.1}$$

Copestake and Briscoe explain that "most idiosyncratic deverbal compounds would be treated as relational." Prepositional compounds are compounds that could easily be paraphrased using a prepositional

TABLE 6.1: The compound noun categories proposed by Copestake and Briscoe (2005).

Compound category	Examples
Listed	"home secretary"
Hypernymic	"tuna fish", "oak tree"
Deverbal	"word inflection", "machine operation"
Relational	"jazz fan", "football supporter"
Made-of	"steel sword", "cardboard box"
Prepositional	"airshow accident"
Non-deverbal verb	"steel town", "cotton town"
Non-paraphrasable	"listeria society"

phrase after the first noun, as in "accident *at* (an/the) airshow". The non-deverbal verb compounds are compounds that are not easily paraphrasable using a preposition. For example, "steel town" can be paraphrased as "town *producing* steel". While their last category, of non-paraphrasable compounds, includes pragmatic compounds, such as the one shown in their example.

Girju et al. (2005) propose 35 different semantic relations for interpreting and classifying compound nouns. For example, they propose the semantic relations *part-whole* and *is-a (hypernymy)*, as in "girl mouth" and "Dallas city", respectively.

For endocentric compounds, the whole compound is a hyponym of the compound's grammatical head. Some Norwegian examples of this phenomenon are «*skolisse*» ("shoe lace"), «*bjørnejakt*» ("bear hunt"), and «*eiendomsskatt*» ("property tax"). However, these are slightly different from what Copestake and Briscoe (2005) refer to as hypernymic and Girju et al. (2005) classify with their *is-a (hypernymy)* relation, because the left constituents are not hyponyms of the right constituents. On the other hand, as shown in Figure 6.1, «*jazzmusikk*» ("jazz music") is a hypernymic, because as the figure shows, `music_n_1` is an (indirect) hypernym of `jazz_n_2`, according to WordNet.

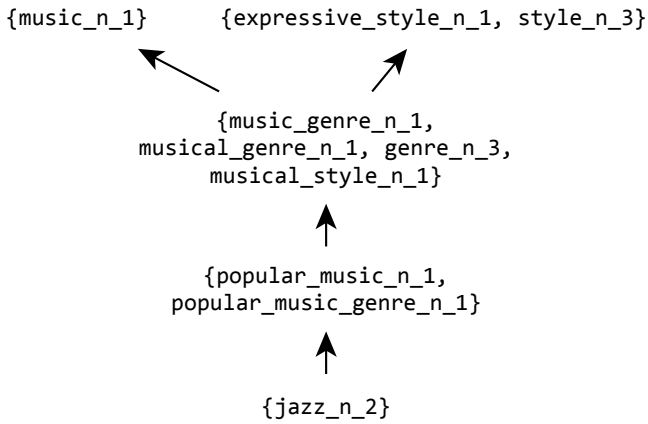


FIGURE 6.1: Hypernyms of "jazz" found in WordNet.

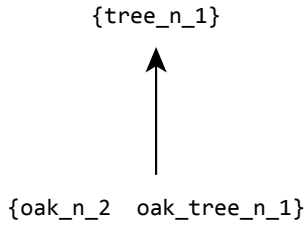


FIGURE 6.2: Hypernym of oak_tree_n_1 as defined by WordNet.

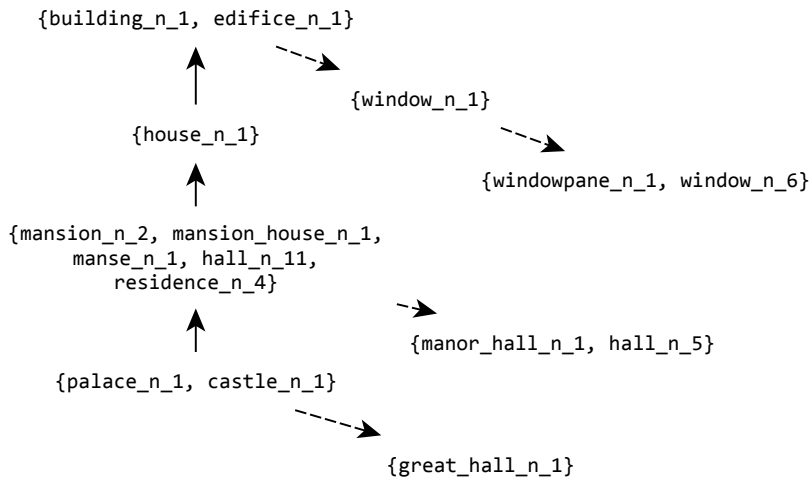


FIGURE 6.3: Hypernyms of "castle" and part holonyms for some of them. Hypernymy edges are solid, while holonym edges are stippled.

Ordnett could, for example, be used to evaluate and rank valid compound-word analyses returned from a compound-word analyzer, with respect to these two last-mentioned semantic relationships. For example, Ordnett contains the mappings

$$\text{ordnett}(\textit{tre}) = \{\texttt{tread\_v\_1}, \texttt{tree\_diagram\_n\_1}, \texttt{three\_adj\_1}, \texttt{wood\_n\_2}, \texttt{tree\_n\_1}\} \tag{6.2}$$

and

$$\text{ordnett}(\textit{eik}) = \{\texttt{quercus\_n\_1}, \texttt{oak\_tree\_n\_1}\}, \tag{6.3}$$

and, as shown in Figure 6.2, in accordance with WordNet, `tree_n_1` is a hypernym of `oak_tree_n_1`.

Furthermore, when analyzing the example compounds presented in the article by Johannessen and Hauglin (1996), the automatic compound analyzer failed in ranking the candidates of «*slottsvinduene*» ("castle windows.the") correctly; the—in principle valid—interpretation «*slottsvin-duene*» ("castle wine-doves.the") was ranked higher. However,

the candidate analysis ranked second was «*slott-s-vinduene*» ("castle windows.the"). Now, as Ordnett includes the mappings

$$\text{ordnett}(\textit{slott}) = \{\texttt{manor\_house\_n\_1}, \texttt{hall\_n\_11}, \texttt{chateau\_n\_1}, \texttt{castle\_n\_1}, \texttt{palace\_n\_1}\} \tag{6.4}$$

and

$$\text{ordnett}(\textit{vinduene}) = \{\texttt{window\_n\_1}, \texttt{fenestra\_n\_1}, \texttt{windowpane\_n\_1}\}, \tag{6.5}$$

and as shown in Figure 6.3, actually all of the senses that ordnett(*slott*) maps to are hyponyms of `building_n_1`, which in turn has both `window_n_1` and `windowpane_n_1` as (indirect) part holonyms, also known as a part-whole relation.

Even though «*eiketre*» ("oak tree") may be considered a lexicalized compound, the above examples show how an automatic compound-word analyzer may exploit information in Ordnett to improve the ranking of candidate analyses by prioritizing analyses that satisfy either the hypernymic or part-whole relationship.

6.2 Ontology Alignment

At first sight, it may seem like a straightforward task to use the output from Verto to create a Norwegian ontology aligned with WordNet. However, there are a few difficulties with such an approach.

Simple, noncompound Norwegian words should be quite easy to map, as long as possible mappings to WordNet are found. If so, each sense of the Norwegian word would be mapped to its corresponding sense in WordNet.

However, a problem arises when it comes to mapping Norwegian compounds when there is no single WordNet concept covering the compound meaning. For example, the sense of «*bjørnejakt*» that translates into (`bear_n_1`, `hunt_n_5`) is a kind of hunt, but it is a hunt where bears are the target. One way of solving this would be to place «*bjørnejakt*» as a new hyponym concept of `hunt_n_5` (or as a reference from the Norwegian wordnet node to an anonymous concept in the same place).

Another, related problem can be observed with verbs. For example, the verb meaning of the Norwegian «*skatt*» translates into (`pay_v_1`, `tax_n_1`). And though `pay_v_1` is a kind of paying, the second part of the translation is really defining a selectional restriction.

6.3 Answers to the Research Questions

In this section I will provide answers to the research questions presented in Section 1.3, based on the findings so far.

Answers to the Research Questions

Q1 Is it possible to develop a method for automatically building a broad-domain, general semantic resource for Norwegian that is compatible with existing freely available, widely used, English semantic resources?

Answer: Yes. Verto, the implementation of the novel method and algorithms presented in Chapter 4, was successfully used to automatically create Ordnett, a broad-domain, general lexical-semantic resource that maps Norwegian words and phrases to concepts in the Princeton WordNet.

As discussed in Section 2.4, others (Knight 1993; Knight and Luk 1994; Okumura and Hovy 1994; Rigau and Agirre 1995; Atserias et al. 1997; Farreres et al. 1998; Vossen 1998; Alonge et al. 1998; Farreres et al. 2002; Tufiş et al. 2004) have created mappings from other non-English languages to WordNet, but, contrary to their methods, my method exploits the implicit information made available through inverse translating the target senses.

Furthermore, as shown in Section 2.4.5, Dyvik (2004) and Nygaard (2006) have also generated broad-domain semantic resources for Norwegian. However, their resources are not in any way aligned or integrated with WordNet, and therefore cannot readily benefit from third-party resources that integrates with or extends WordNet. Additionally, their generated resources provide fewer semantic relations than provided by WordNet and thereby by Ordnett. Finally, in contrast to Nygaard's approach, Verto correctly detects and distinguishes between the different senses of a Norwegian word.

Q2 What restrictions apply to the method? What resources are prerequisite?

Answer: The method described herein requires a) a simple bilingual dictionary that contains information about the lexical category of entries and maps words in the source language to words in the target language; and b) a lexical database that contains semantic information about synonymy, hypernymy, and similarity between senses in the target language. To handle compound words, the method also requires the availability of an automatic compound analyzer.

Q3 For each of the lexical categories nouns, verbs, adjectives, and adverbs, how large fraction of the words in each class does the method work for?

Answer: With the configuration of the mapping framework that provided the maximum average precision value of 0.921 (achieved in Test Run 4), the number of successfully mapped words per lexical category were noun 22,888 (50.3 %); adjective 4,387 (47.7 %); verb 3,495 (65.2 %); and adverb 186 (31.9 %).

With the configuration that provided the maximum recall value of 0.365 (achieved in Test Run 12), the number of successfully mapped words per lexical category were noun 25,879 (56.3 %); adjective 5,357 (56.2 %); verb 4,264 (78.5 %); and adverb 492 (35.3 %). The same configuration also provided the maximum $F_{0.5}$-score of 0.680.

Q4 How well does the method avoid mapping words in the source language to senses in the target language that carry meanings that are not covered by the source words?

Answer: Comparison of the results produced by the system presented herein with those produced by a human expert showed that with a configuration that maximizes the precision value, the system reaches a precision score of 0.921.

This shows that the system is able to rather precisely avoid unwarranted mappings.

Q5 Can the method handle (single word) compounds that occur frequently in Norwegian, but more seldom in English?

Answer: Yes. The framework that implements the method presented herein handles mapping of compounds by applying an integrated compound-word analyzer.

Q6 How may the resulting semantic resource be useful for different areas of Natural Language Processing (NLP) research and development?

Answer: One of the most important benefits of the method is that it produces a semantic resource that is linked to the Princeton WordNet, and thereby can benefit from other semantic resources that extends or complements—but are still compatible with—WordNet.

In this dissertation Verto, an implementation of the method, was used to generate Ordnett, an open-domain, general lexical semantic resource for Norwegian. It should be possible to utilize Ordnett for Norwegian (versions of) applications within most areas of NLP where ontologies are already used. These areas include, but is not necessarily limited to, Word-Sense Disambiguation (WSD), Text Summarization, Document Clustering, Information Extraction (IE), Information Retrieval (IR), Text Interpretation, and Natural Language Generation (NLG).

Q7 Can the method be applied to other languages, and if so, to which ones?

Answer: I see no reason why the method should not be applicable to at least other Germanic languages, like the Scandinavian languages, German, and Dutch.

Contribution

However, if the language makes extensive use of compounding, an automatic compound-word analyzer may be required.

6.4 Contribution

In this dissertation I have presented (1) a method I have developed that consists of a model and several algorithms for automatically mapping content words from a non-English source language to WordNet senses. Reflecting the importance of compounding in Norwegian, the method is able to handle compounds. Therefore, I also presented (2) a practical implementation, including algorithms and a grammar, of a program for automatically analyzing Norwegian compounds.

Furthermore, I showed (3) that Verto, an implementation of the model and algorithms, was used to create Ordnett, the first large-scale, open-domain lexical-semantic resource for Norwegian with a rich number of semantic relations. Because Ordnett inherits all the semantic relations from WordNet, it provides access to many more semantic relations than any of the preexisting approaches to generate general Norwegian lexical-semantic resources, presented by Nygaard (2006) and Dyvik (2004). Because Ordnett is a mapping of Norwegian words onto WordNet senses, the resource can also utilize other semantic resources that integrate with WordNet to extend or complement it. Examples of such resources include VerbNet (Kipper et al. 2000b, 2004) and FrameNet (Fillmore et al. 2003; Baker et al. 2003). This is another feature not provided by the other Norwegian approaches. Additionally, because new versions of Ordnett can easily be generated by rerunning Verto, costs related to maintenance and upgrades of the resource are strongly reduced, compared to manually performing such tasks. Therefore, Ordnett will benefit from any improvement of WordNet.

Finally, I showed (4) how Ordnett can be used in an open-domain question answering (open-domain QA) system, thereby arguing that my method and automatically generated lexical semantic resource makes it possible to build large-scale open-domain Natural Language Understanding (NLU) systems, that offer both wide coverage and deep analyses, for Norwegian texts.

As shown in chapters 4 and 5 the Ordnett resource, created with Verto, constitutes a useful large-scale, lexical-semantic resource for Norwegian with a rich number of semantic relations.

Only the future will show, but I hope that both the methods and the resource presented herein can be of great help to future research on technologies related to—and products for—the Norwegian language. The cultural, technological, economical, and educational consequences caused by the relative scarcity of advanced Norwegian language-technological solutions and resources—compared to the situation for other, larger languages—has already been widely acknowledged (Simonsen 2005; Norwegian Language Council 2005; Norwegian Ministry of Culture and Church Affairs 2008). I hope that my work, presented herein, can help improve the current situation.

6.5 Future Work

There are several possible directions that the research on Ordnett might take in the future.

The approach implemented in Verto makes it easy to automatically generate new versions of Ordnett when new versions of the resources it consumes, like WordNet or bilingual dictionary resources, become available. Similarly, generating new Ordnett versions, of higher quality, if improved mapping methods or algorithms are added to Verto, also requires little effort. Nonetheless, improving the general quality of the generated resources, manifested as higher levels of coverage, recall, or precision, will be of great importance in the future.

On the other hand, keeping the generated resource updated might not be solved by algorithms and newer versions of the input resources alone. For example, a word's senses in a language sometimes change over time. Kokkinakis (2000) reports research on a corpora-based approach for augmenting machine-readable dictionaries with such novel uses of words. Updating Ordnett in a similarly fashion might prove advantageous in the future. In general, supplementing Verto with corpus-based approaches might hold the key to improving Ordnett's coverage.

Based on the preliminary results from the TUClopedia project presented herein, one can also catch a glimpse of the extent of the problems ahead in the development of TUClopedia. Even though large parts of the system has already been finished and integrated with each other, it seems safe to state that most of the work lies ahead, because most of the remaining work deals with increasing the coverage of the NLU parts of the system.

One of the first goals of future TUClopedia development should be to improve the system to the extent that we can provide large-scale empirical results and evaluations of its workings.

In order to do that, the integration of external sources needs to be improved. TUClopedia does not yet make full use of WordNet's features, and implementing better WSD should be a high priority. For example, it could be interesting to see how the conceptual density measure (Rigau and Agirre 1995; Agirre and Rigau 1996) might be applied to improve TUClopedia's WSD process.

Furthermore, the coverage of LexTUC's internal lexical-semantic knowledge should be extended. This might be done by improving the interaction between LexTUC's internal lexical-semantic knowledge and WordNet, or WordNet-compatible resources. For example, it would be a huge step forward if LexTUC's information about selectional restrictions of different verbs could be increased by integrating WordNet-compatible resources, like FrameNet (Baker et al. 2003), VerbNet (Kipper et al. 2000a, 2004), or eXtended WordNet (Moldovan and Novischi 2004).

Developing adequate user interfaces for presenting answers will also be needed. This work could probably make use of a document retrieval system, which is already partly implemented in TUClopedia, but for now precise answers are prioritized.

6.6 Final Words

As *Niels Bohr*[1] stated, "prediction is very difficult, especially about the future." Therefore, I can only hope that parts of my work presented herein will be used in research that eventually finds it way into the language technologies of tomorrow. If so should happen, my work has served as a stepping stone that might eventually help improve some aspects of everyday life affected by the Norwegian language. Or, perhaps, some other non-English languages too.

1 *Niels Bohr* (1885–1962), Danish physicist.

Appendix

Brief Introduction to Graph Theory

A

This appendix provides a short introduction to a small part of graph theory to ensure that the terms used in Sections 4.9 and 4.11 are properly introduced.

A graph G is a tuple (V, E) where V denotes a finite set and E defines a binary relation on V. We call each element of V a vertex (or node) while each element of E represents an edge between two such vertices. Figure A.1 visualizes a graph, where the circles represent vertices and the lines between them represent the edges.

A *path* of length k between two vertices u' and v' in a graph consists of a sequence $\langle v_1, v_2, \ldots, v_k \rangle$ where $(v_{i-1}, v_i) \in E$ holds for all $i = 2, \ldots, k$, and $v_1 = u'$ and $v_k = v'$.

Graphs can be directed or undirected. In a directed graph the binary relation E defined on V consists of ordered pairs, while in undirected graphs, E consists of unordered pairs. The arrowheads in Figure A.2 on the next page identify the direction of each edge.

Furthermore, we usually distinguish between cyclic and acyclic graphs. A cyclic graph contains edges so that one can follow a nonempty path from one vertex in the graph back to itself. For example, the edges $(2, 2)$ and $(5, 1)$ in Figure A.2 yield cycles in the graph. An acyclic graph

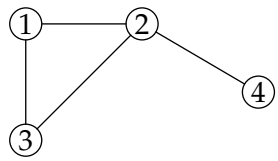


Figure A.1: An *undirected graph* $G = (V, E)$, where $V = \{1, 2, 3, 4\}$ and $E = \{(1, 2), (1, 3), (2, 3), (2, 4)\}$.

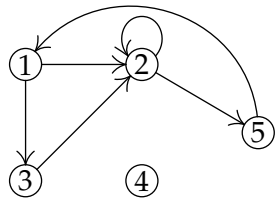


FIGURE A.2: A *directed graph* $G = (V, E)$, where $V = \{1, 2, 3, 4, 5\}$ and $E = \{(1, 2), (1, 3), (2, 2), (2, 5), (3, 2), (5, 1)\}$.

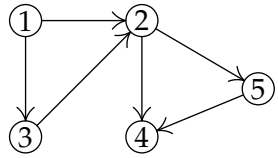


FIGURE A.3: A *directed acyclic graph* $G = (V, E)$, where $V = \{1, 2, 3, 4, 5\}$ and $E = \{(1, 2), (1, 3), (2, 4), (2, 5), (3, 2), (5, 4)\}$.

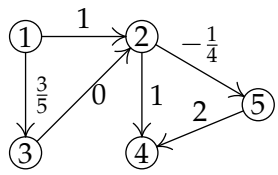


FIGURE A.4: A *weighted directed acyclic graph* $G = (V, E)$, where $V = \{1, 2, 3, 4, 5\}$ and $E = \{(1, 2, 1), (1, 3, \frac{3}{5}), (2, 4, 1), (2, 5, -\frac{1}{4}), (3, 2, 0), (5, 4, 2)\}$.

contains no such cycles. We call a directed and acyclic graph for a directed acyclic graph (DAG), see Figure A.3 on the facing page for an example.

Every DAG contains at least one source and at least one sink. The vertices without incoming edges constitute the sources of a DAG, and likewise, vertices without outgoing edges constitute its sinks. The DAG in Figure A.3 has only one source $\{1\}$ and one sink $\{4\}$.

Graphs can be weighted. In a weighted graph, every edge has a certain weight. Likewise, in a *weighted* DAG, $G = (V, E)$, every edge is defined as a triple, (u, v, w), where u and v represent the source and target vertices respectively, while w represents the weight of that particular edge. For example, in Figure A.4 on the preceding page the edge $(2, 5, -\frac{1}{4})$ has a weight of $-\frac{1}{4}$, while the edge $(3, 2, 0)$ has a weight of 0.

For a more thorough introduction to graph theory, please consult, for example, (Cormen et al. 2001; Appendix B) or (West 2001).

References

The numbers following each entry in the bibliography denote the pages where that work was referenced.

Agirre, Eneko, Olatz Ansa, Eduard H. Hovy, and David Martínez. 2000. Enriching very large ontologies using the WWW. In *Proceedings of the First Workshop on Ontology Learning OL'2000*. Berlin, Germany. Held in conjunction with the 14th European Conference on Artificial Intelligence ECAI'2000. 81

Agirre, Eneko, Xabier Arregi, Xabier Artola, Arantza Díaz de Ilarraza, and Kepa Sarasola. 1994. Conceptual Distance and automatic spelling correction. In *Proceedings of the Workshop on Computational Linguistics for Speech and Handwriting Recognition*. Leeds, UK. 27

Agirre, Eneko, and German Rigau. 1996. Word sense disambiguation using Conceptual Density. In *Proceedings of the 16th Conference on Computational Linguistics*, 16–22. Morristown, NJ, USA: Association for Computational Linguistics. 25, 26, 196

Alonge, Antonietta, Nicoletta Calzolari, Piek Vossen, Laura Bloksma, Irene Castellon, Maria Antonia Marti, and Wim Peters. 1998. The linguistic design of the EuroWordNet database. *Computers and the Humanities* 32:91–115. 24, 192

Alvar Ezquerra, Manuel, ed. 1987. *Diccionario General Ilustrado de la Lengua Española*. Barcelona, Spain: Vox Biblograf S.A. 27

Amble, Tore. 2000. BusTUC - a natural language bus route oracle. In *Proceedings of the 6th Applied Natural Language Processing Conference*, 1–6. Seattle, Washington, USA: Association for Computational Linguistics. 4, 10, 15, 35, 37, 38, 39, 152

———. 2003. *The understanding computer: Natural language understanding in practice*. Trondheim, Norway: Department of Computer and

Information Science, Norwegian University of Science and Technology. Preliminary version. xxv, 4, 6, 39

Amble, Tore, Martin Thorsen Ranang, and Rune Sætre. 2002. *The understanding computer: A tutorial*. Department of Computer and Information Science, Norwegian University of Science and Technology, Trondheim, Norway. 37

Ashburner, Michael, Catherine A. Ball, Judith A. Blake, David Botstein, Heather Butler, J. Michael Cherry, Allan P. Davis, Kara Dolinski, Selina S. Dwight, Janan T. Eppig, Midori A. Harris, David P. Hill, Laurie Issel-Tarver, Andrew Kasarskis, Suzanna Lewis, John C. Matese, Joel E. Richardson, Martin Ringwald, Gerald M. Rubin, and Gavin Sherlock. 2000. Gene Ontology: tool for the unification of biology. *Nature Genetics* 25:25–29. 10

Atserias, Jordi, Salvador Climent, Javier Farreres, German Rigau, and Horacio Rodríguez. 1997. Combining multiple methods for the automatic construction of multilingual WordNets. In *Proceedings of the International Conference on Recent Advances in Natural Language Processing (RANLP 97)*. Tzigov Chark, Bulgaria. 27, 28, 29, 30, 31, 147, 192

Ayad, Hanan, and Mohamed Kamel. 2002. Topic discovery from text using aggregation of different clustering methods. In *Advances in Artificial Intelligence: 15th Conference of the Canadian Society for Computational Studies of Intelligence, AI 2002, Calgary, Canada, May 27–29, 2002, proceedings*, ed. Robin Cohen and Bruce Spencer, vol. 2338 of *Lecture Notes in Artificial Intelligence*, 161–175. Berlin, Heidelberg: Springer-Verlag. 14

Baker, Collin F., Charles J. Fillmore, and Beau Cronin. 2003. The structure of the FrameNet database. *International Journal of Lexicography* 16(3):281–296. 81, 183, 194, 196

Bateman, John A. 1990. Upper modeling: organizing knowledge for natural language processing. In *Proceedings of the Fifth International Workshop on Natural Language Generation*. Pittsburgh, PA, USA. 24

Bird, Steven. 2006. NLTK: The Natural Language Toolkit. In *Proceedings of the COLING/ACL Interactive Presentation Sessions*, 69–72. Sydney, Australia: Association for Computational Linguistics. 57

Bird, Steven, Ewan Klein, and Edward Loper. 2008. Natural Language Processing in Python. Available on-line, `http://nltk.org/index.php/Book`. Accessed Jan. 14, 2008. 57

Bird, Steven, and Edward Loper. 2004. NLTK: The Natural Language Toolkit. In *Proceedings of the ACL 2004 Interactive Poster and Demonstration Sessions*, 31. Barcelona, Spain: Association for Computational Linguistics. 57

Bruland, Tore. 2002. ExamTUC - a simple examination system in natural language. Sivilingeniør's thesis, Department of Computer and Information Science, Norwegian University of Science and Technology, Trondheim, Norway. 4, 38

Chandrasekaran, B., J. R. Josephson, and V. R. Benjamins. 1999. What are ontologies, and why do we need them? *Intelligent Systems and Their Applications, IEEE* 14:20–26. 10

Chen, Yu, Andreas Eisele, and Martin Kay. 2008. Improving Statistical Machine Translation Efficiency by Triangulation. In *Proceedings of the Sixth International Conference on Language Resources and Evaluation (LREC 2008)*, 2875–2880. Marrakech, Morocco. 33

Collins, Henry H., Jr., and Colin Smith, eds. 1971. *Collins Spanish-English/English-Spanish Dictionary*. New York, USA: William Collins Sons & Co. Ltd. 24, 31

Constantinescu-Fulöp, Zoran, and Jörg Cassens. 2003. It's magic: SourceMage GNU/Linux as HPC cluster OS. In *LinuxTag 2003*. Karlsruhe, Germany: LinuxTag. 181

Converse, Tim, Ronald M. Kaplan, Barney Pell, Scott Prevost, Lorenzo Thione, and Chad Walters. 2008. Powerset's natural language Wikipedia search engine. In *Wikipedia and Artificial Intelligence: An Evolving Synergy: Papers from the AAAI Workshop*, 67. Techical Report WS-08-15, Menlo Park, California, USA: The AAAI Press. 1

Copestake, Ann, and John Edward Briscoe. 2005. Noun compounds revisited. In *Charting a New Course: Natural Language Processing and Information Retrieval. Essays in Honour of Karen Spärck Jones*, ed. John I. Tait, vol. 16 of *The Information Retrieval Series*, chap. 9, 129–154. Dordrecht, The Netherlands: Springer. xv, 187, 188

Copestake, Ann, Dan Flickinger, Carl Pollard, and Ivan Sag. 2005. Minimal Recursion Semantics: An introduction. *Research on Language & Computation* 3:281–332. 40

Cormen, Thomas H., Charles E. Leiserson, Ronald L. Rivest, and Clifford Stein. 2001. *Introduction to algorithms*. 2nd ed. MIT Electrical and

Computer Science Series, Cambridge, Massachusetts, USA: The MIT Press. 201

Cruse, D. A. 1986. *Lexical semantics*. Cambridge textbooks in linguistics, Cambridge, UK: Cambridge University Press. 18

Dorr, Bonnie Jean. 1992. The use of lexical semantics in interlingual machine translation. *Machine Translation* 7(3):135–193. 32

———. 1993. Interlingual machine translation: a parameterized approach. *Artificial Intelligence* 63(1–2):429–492. 32

———. 1997. Large-scale dictionary construction for foreign language tutoring and interlingual machine translation. *Machine Translation* 12(4):271–325. 32, 33

Dorr, Bonnie Jean, Joseph Garman, and Amy Weinberg. 1995. From syntactic encodings to thematic roles: Building lexical entries for interlingual MT. *Machine Translation* 9(3):71–100. 33

Dyvik, Helge. 1999. The universality of f-structure: discovery or stipulation? The case of modals. In *Proceedings of the LFG '99 Conference*, ed. Miriam Butt and Tracy Holloway King. The University of Manchester: CSLI Publications. 40

———. 2004. Translations as semantic mirrors. From parallel corpus to WordNet. *Language and Computers* 49(1):311–326. 2, 4, 34, 192, 194

———. 2005. Translations as a semantic knowledge source. In *Proceedings of the Second Baltic Conference on Human Language Technologies*. Tallinn: Institute of Cybernetics at Tallinn University of Technology, Institute of the Estonian Language. 144, 145

Earley, Jay. 1970. An efficient context-free parsing algorithm. *Communications of the ACM* 13(2):94–102. 57

van Els, Theo J.M. 2001. The European Union, its Institutions and its Languages: Some Language Political Observations. *Current Issues In Language Planning* 2(4):311–360. 1

Faarlund, Jan Terje, Svein Lie, and Kjell Ivar Vannebo. 1997. *Norsk referansegrammatikk*. Oslo, Norway: Universitetsforlaget. xxiii, 152, 153

Farreres, Javier, German Rigau, and Horacio Rodríguez. 1998. Using WordNet for building WordNets. In *Use of WordNet in Natural Language Processing Systems: Proceedings of the Conference*, ed. Sanda M. Harabagiu, 65–72. Somerset, New Jersey: Association for Computational Linguistics. 31, 192

Farreres, Javier, Horacio Rodríguez, and Karina Gibert. 2002. Semiautomatic creation of taxonomies. In *Coling-02 on semanet*, 1–7. Morristown, NJ, USA: Association for Computational Linguistics. 31, 147, 192

Fellbaum, Christiane. 1998a. A semantic network of English: The mother of all WordNets. *Computers and the Humanities* 32:209–220. 77

———. 1998b. A semantic network of English verbs. In Fellbaum (1998c), chap. 3, 69–104. 20, 21

Fellbaum, Christiane, ed. 1998c. *WordNet: An electronic lexical database*. Language, Speech, and Communication, Cambridge, Massachusetts, USA: The MIT Press. xxv, 4, 6, 11, 76, 77, 207, 211, 213, 216, 218

Fillmore, Charles J., Christopher R. Johnson, and Miriam R.L. Petruck. 2003. Background to FrameNet. *International Journal of Lexicography* 16(3):235–250. `http://ijl.oxfordjournals.org/cgi/reprint/16/3/235.pdf`. 81, 183, 194

Flickinger, Dan. 2000. On building a more efficient grammar by exploiting types. *Natural Language Engineering* 6(1):15–28. 40

Gambäck, Björn, and Stefan Ljung. 1993. Question Answering in the Swedish Core Language Engine. In *Proceedings of the 4th Scandinavian Conference on Artificial Intelligence*, ed. Erik Sandewall and Carl Gustaf Jansson, 212–225. Frontiers in Artificial Intelligence and Applications 18, Stockholm, Sweden: IOS Press, Amsterdam, Holland. 35, 37

Gamut, L.T.F. 1991a. *Intensional logic and logical grammar*, vol. 2 of *Logic, Language, and Meaning*. Chicago and London: The University of Chicago Press. 167

———. 1991b. *Introduction to logic*, vol. 1 of *Logic, Language, and Meaning*. Chicago and London: The University of Chicago Press. 18

Gangemi, Aldo, Nicola Guarino, and Alessandro Oltramari. 2001. Conceptual analysis of lexical taxonomies: the case of WordNet top-level.

In *Proceedings of the International Conference on Formal Ontology in Information Systems*, 285–296. ACM Press. 81

Girju, Roxana, Dan I. Moldovan, Marta Tatu, and Daniel Antohe. 2005. On the semantics of noun compounds. *Computer Speech & Language* 19:479–496. 188

Gomez, Fernando. 1994. Knowledge acquisition from real-world texts: some lessons learned. In *Proceedings of the Sixth International Conference on Tools with Artificial Intelligence*, 229–230. 15, 37

Gomez, Fernando, Richard D. Hull, and Carlos Segami. 1994. Acquiring knowledge from encyclopedic texts. In *Proceedings of the 4th Conference on Applied Natural Language Processing*, 84–90. Association for Computational Linguistics, Stuttgart, Germany: Morgan Kaufmann Publishers Inc. 15

Gonzalo, Julio, Felisa Verdejo, Carol Peters, and Nicoletta Calzolari. 1998. Applying EuroWordNet to cross-language text retrieval. *Computers and the Humanities* 32:185–207. 15

Green, Claude Cordell, and Bertram Raphael. 1968. The use of theorem-proving techniques in question-answering systems. In *Proceedings of the 1968 23rd ACM National Conference*, 169–181. New York, NY, USA: ACM. 34

Grishman, Ralph. 2003. Information extraction. In *The Oxford Handbook of Computational Linguistics*, ed. Ruslan Mitkov. Oxford, UK: Oxford University Press. 14

Grunfeld, Laszlo, and Kui-Lam Kwok. 2006. Sentence ranking using keywords and meta-keywords. In Strzalkowski and Harabagiu (2006), 229–258. 35, 149

Hamburger, Henry, and Dana Richards. 2002. *Logic and Language Models for Computer Science*, chap. 10, 201–222. Upper Saddle River, New Jersey, USA: Prentice Hall. 70

Harabagiu, Sanda M., George A. Miller, and Dan I. Moldovan. 1999. WordNet 2 - a morphologically and semantically enhanced resource. In *Proceedings of SIGLEX99: Standardizing Lexical Resources*, 1–8. Special Interest Group on the Lexicon of the Association for Computational Linguistics and the National Science Foundation, University of Maryland, College Park, Maryland, USA: Association for Computational Linguistics. 80, 152

Hartmann, R. R. K., and Gregory James. 2002. *Dictionary of lexicography*. Routledge. 92

Haslerud, Vibecke C. D., and Petter Henriksen, eds. 2003. *Engelsk stor ordbok: Engelsk-norsk / norsk-engelsk*. Oslo, Norway: Kunnskapsforlaget. 72, 74, 76, 152, 153

Hasselgård, Hilde, Stig Johansson, and Per Lysvåg. 1998. *English grammar: Theory and use*. Oslo, Norway: Universitetsforlaget. 18, 20

Hellan, Lars, and Petter Haugereid. 2003. The NorSource grammar–an excercise in the Matrix Grammar building design. *Proceedings of Workshop on Multilingual Grammar Engineering, ESSLLI 2003*. 40

Henriksen, Petter, ed. 2003. *Aschehougs og Gyldendals Store Norske Leksikon*. Oslo, Norway: Kunnskapsforlaget. 149, 151, 152, 153, 163, 167, 179

Hobbs, Jerry R., Mark E. Stickel, Douglas E. Appelt, and Paul Martin. 1993. Interpretation as abduction. *Artificial Intelligence* 63(1–2):69–142. 15

Hotho, Andreas, Steffen Staab, and Gerd Stumme. 2003. Ontologies improve text document clustering. In *Proceedings of the Third IEEE International Conference on Data Mining (ICDM'03)*, ed. Xindong Wu, Alex Tuzhilin, and Jude Shavlik, 541–544. Melbourne, Florida, USA: IEEE Computer Society, Los Alamitos, CA, USA. 14

Hull, Richard D., and Fernando Gomez. 1999. Automatic acquisition of biographic knowledge from encyclopedic texts. *Expert Systems with Applications* 16(3):261–270. 15, 37, 153

Jackendoff, Ray S. 1992. *Semantic structures*, vol. 18 of *Current Studies in Linguistics*. 4th ed. Cambridge, Massachusetts, USA: The MIT Press. 32

Jacobs, Paul S., and Lisa F. Rau. 1993. Innovations in text interpretation. *Artificial Intelligence* 63(1–2):143–191. 15

Jing, Hongyan, and Kathleen McKeown. 1998. Combining multiple, large-scale resources in a reusable lexicon for natural language generation. In *Proceedings of the 17th International Conference on Computational Linguistics*, 607–613. Morristown, NJ, USA: Association for Computational Linguistics. 16

Joachims, Thorsten. 1997. A probabilistic analysis of the Rocchio algorithm with TFIDF for text categorization. In *International Conference on Machine Learning (ICML)*. Nashville, Tennessee, USA. 14

Johannessen, Janne Bondi. 2001. Sammensatte ord. *Norsk Lingvistisk Tidsskrift* 19:59–91. 22, 23, 24, 41, 60, 62

Johannessen, Janne Bondi, and Helge Hauglin. 1996. An automatic analysis of Norwegian compounds. In *Papers from the 16th Scandinavian Conference of Linguistics*. Turku/Åbo, Finland. 23, 24, 42, 57, 58, 66, 67, 68, 70, 190

Joseph, John Earl. 2004. *Language and Identity: National, Ethnic, Religious*. Hampshire, UK: Palgrave Macmillan. 1

Jurafsky, Daniel Saul, and James H. Martin. 2000. *Speech and Language Processing*. Prentice Hall Series in Artificial Intelligence, Upper Saddle River, New Jersey, USA: Prentice Hall. 57, 77, 110, 111

Jönsson, Arne, Frida Andén, Lars Degerstedt, Annika Flycht-Eriksson, Magnus Merkel, and Sara Norberg. 2004. Experiences from combining dialogue system development with information extraction techniques. In *New Directions in Question Answering*, ed. Mark T. Maybury, 153–168. AAAI/MIT Press. 10, 37

Karlsson, Fred. 1992. SWETWOL: A Comprehensive Morphological Analyser for Swedish. *Nordic Journal of Linguistics* 15(01):1–45. 23, 50

Kay, Martin. 1997. The proper place of men and machines in language translation. *Machine Translation* 12(1):3–23. 33

Kipper, Karin, Hoa Trang Dang, and Martha Palmer. 2000a. Class-based construction of a verb lexicon. In *Seventeenth National Conference on Artificial Intelligence (AAAI-2000)*. Austin, Texas, USA. 81, 183, 196

Kipper, Karin, Hoa Trang Dang, William Schuler, and Martha Palmer. 2000b. Building a class-based verb lexicon using TAGs. In *TAG+5 Fifth International Workshop on Tree Adjoining Grammars and Related Formalisms*. Paris, France. 194

Kipper, Karin, Benjamin Snyder, and Martha Palmer. 2004. Extending a verb-lexicon using a semantically annotated corpus. In *Proceedings of the Fourth International Conference on Language Resources and Evaluation (LREC 2004)*. Lisbon, Portugal. 81, 183, 194, 196

Knight, Kevin. 1993. Building a large ontology for machine translation. In *HLT '93: Proceedings of the Workshop on Human Language Technology*, 185–190. Morristown, NJ, USA: Association for Computational Linguistics. 24, 192

Knight, Kevin, and Steve K. Luk. 1994. Building a large-scale knowledge base for machine translation. *Proceedings of the Twelfth National Conference on Artificial Intelligence* 1:773–778. 11, 15, 24, 31, 83, 192

Kohl, Karen T., Douglas A. Jones, Robert C. Berwick, and Naoyuki Nomura. 1998. Representing verb alternations in WordNet. In Fellbaum (1998c), chap. 6, 153–178. 184

Kokkinakis, Dimitrios. 2000. Concordancing revised *or* how to aid the recognition of new senses in very large corpora. In *Proceedings of the Second International Conference on Natural Language Processing: NLP 2000*, ed. Dimitris N. Christodoulakis, vol. 1835 of *Lecture Notes in Artificial Intelligence*, 370–381. Patras, Greece: Springer-Verlag, Berlin, Heidelberg. 195

Kokkinakis, Dimitrios, Maria Toporowska Gronostaj, and Karin Warmenius. 2000. Annotating, Disambiguating & Automatically Extending the Coverage of the Swedish SIMPLE Lexicon. In *Proceedings of the Second Conference on Language Resources and Evaluation (LREC-2000)*. Athens, Greece. 23, 31, 67

Kowalski, Robert, and Marek Sergot. 1986. A logic-based calculus of events. *New Generation Computing* 4:67–95. 39

Kupiec, Julian. 1993. Murax: a robust linguistic approach for question answering using an on-line encyclopedia. In *Proceedings of the Sixteenth Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, 181–190. ACM Press. 37

Kwon, Namhee, and Eduard H. Hovy. 2006. Integrating semantic frames from multiple sources. In *Proceedings of the Seventh International Conference on Intelligent Text Processing and Computational Linguistics (CICLing)*, vol. 3878 of *Lecture Notes in Computer Science*, 1–12. Mexico City, Mexico: Springer-Verlag, Berlin, Heidelberg. 183

Lambert, Wallace E. 1972. *Language, psychology, and culture*. Language Science and National Development Series, Stanford, California, USA: Stanford University Press. 1

Larsen, Bjornar, and Chinatsu Aone. 1999. Fast and effective text mining using linear-time document clustering. In *KDD '99: Proceedings of the Fifth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 16–22. New York, NY, USA: ACM. 14

Lauer, Mark. 1995. Designing statistical language learners: Experiments on noun compounds. Ph.D. thesis, Macquarie University, Sydney, Australia. 187

Laver, John, and Jan Roukens. 1996. The Global Information Society and Europe's Linguistic and Cultural Heritage. In *Language, Culture and Communication in Contemporary Europe*, ed. Charlotte Hoffmann, 1–27. Bristol, UK: Multilingual Matters. 1

Lenat, Douglas B. 1995. CYC: A large-scale investment in knowledge infrastructure. *Communications of the ACM* 38(11):33–38. 11

———. 2006. *Computers versus Common Sense*. Google TechTalks, May 30. 12 min., 9 sec. AVI, `http://video.google.com/videoplay?docid=-7704388615049492068` (accessed November 4, 2008). 36

Lenci, Alessandro, Nuria Bel, Federica Busa, Nicoletta Calzolari, Elisabetta Gola, Monica Monachini, Antoine Ogonowski, Ivonne Peters, Wim Peters, Nilda Ruimy, Marta Villegas, and Antonio Zampolli. 2000. SIMPLE: A general framework for the development of multilingual lexicons. *International Journal of Lexicography* 13(4):249–263. 23, 25, 34

Levin, Beth. 1993. *English verb classes and alternations: A preliminary investigation*. The University of Chicago Press. 33

Lewis, Harry R., and Christos H. Papadimitriou. 1998. *Elements of the Theory of Computation*, chap. 2, 55–112. 2nd ed. Upper Saddle River, New Jersey, USA: Prentice Hall. 70

Mahesh, Kavi, and Sergei Nirenburg. 1996. Meaning representation for knowledge sharing in practical machine translation. In *Proceedings of Florida Artificial Intelligence Research Symposium, FLAIRS-96, Special Track on Information Interchange*. Key West, FL, USA. 15

Mani, Inderjeet, and Mark T. Maybury, eds. 1999. *Advances in Automatic Text Summarization*. Cambridge, Massachusetts, USA: The MIT Press. 13

Manning, Christopher D., Prabhakar Raghavan, and Hinrich Schütze. 2008. *Introduction to Information Retrieval*. New York, NY, USA: Cambridge University Press. 14

Matsumoto, Makoto, and Takuji Nishimura. 1998. Mersenne twister: a 623-dimensionally equidistributed uniform pseudo-random number generator. *ACM Transactions on Modeling and Computer Simulation (TOMACS)* 8(1):3–30. 111

McKeown, Kathleen, Jacques Robin, and Karen Kukich. 1995. Generating concise natural language summaries. *Information Processing & Management* 31(5):703–733. 13

Miller, George A. 1985. Dictionaries of the mind. In *Proceedings of the 23rd Annual Meeting of the Association for Computational Linguistics*, 305–314. Morristown, NJ, USA: Association for Computational Linguistics. 76, 77

———. 1995. WordNet: a lexical database for English. *Communications of the ACM* 38(11):39–41. xxv, 4, 6

———. 1998. Nouns in WordNet. In Fellbaum (1998c), chap. 2, 23–46. 18

Miller, George A., Richard Beckwith, Christiane Fellbaum, Derek Gross, and Katherine J. Miller. 1990. Introduction to WordNet: an on-line lexical database. *International Journal of Lexicography* 3(4):235–244. (Revised August 1993). 76

Miller, George A., and Christiane Fellbaum. 2007. Wordnet then and now. *Language Resources and Evaluation* 41:209–214. 153

Miller, George A., and Florentina Hristea. 2006. WordNet nouns: Classes and instances. *Computational Linguistics* 32(1):1–3. 76, 81

Moldovan, Dan I., Christine Clark, Sanda M. Harabagiu, and Steve Maiorano. 2003. COGEX: a logic prover for question answering. In *HLT-NAACL '03: Proceedings of the 2003 Human Language Technology Conference of the North American Chapter of the Association for Computational Linguistics*, 87–93. Edmonton, Canada: Association for Computational Linguistics. 35

Moldovan, Dan I., and Rada Mihalcea. 2000. Using WordNet and lexical operators to improve Internet searches. *Internet Computing, IEEE* 4: 34–43. 15

Moldovan, Dan I., and Adrian Novischi. 2004. Word sense disambiguation of WordNet glosses. *Computer Speech & Language* 18:301–317. 13, 196

Moldovan, Dan I., Marius A. Paşca, and Mihai Surdeanu. 2006. Some advanced features of LCC's Poweranswer. In Strzalkowski and Harabagiu (2006), 3–34. 35

Moldovan, Dan I., and Vasile Rus. 2001. Logic form transformation of WordNet and its applicability to question answering. In *ACL '01: Proceedings of the 39th Annual Meeting of the Association for Computational Linguistics*, 402–409. Morristown, NJ, USA: Association for Computational Linguistics. 35

Navarro, Gonzalo. 2001. A guided tour to approximate string matching. *ACM Computing Surveys (CSUR)* 33(1):31–88. 157

Nirenburg, Sergei, and Victor Raskin. 2004. *Ontological semantics*. Language, Speech, and Communication, Cambridge, Massachusetts, USA: The MIT Press. 1, 9

Nordgård, Torbjørn. 1998. Norwegian computational lexicon (NorKompLeks). In *Proceedings of the 11th Nordic Conference on Computational Linguistics*, ed. Bente Maegaard, 34–44. University of Copenhagen, Denmark: Center for Sprogteknologi. 41, 152, 153

Nordgård, Torbjørn, Martin Thorsen Ranang, and Jostein Ven. 2005. An approach to automatic text production in electronic medical record systems. In *Proceedings of the 9th International Conference on Knowledge-Based Intelligent Information and Engineering Systems (KES 2005)*, ed. Rajiv Khosla, Robert J. Howlett, and Lakhmi C. Jain, vol. 3683 of *Lecture Notes in Artificial Intelligence*, 1187–1194. Melbourne, Australia: Springer-Verlag, Berlin, Heidelberg. 8, 16

Norris, Pippa. 2001. *Digital divide: Civic engagement, information poverty, and the internet worldwide*. Communication, Society and Politics, Cambridge, New York, USA: Cambridge University Press. 150

Norwegian Language Council. 2005. *Norsk i hundre! Norsk som nasjonalspråk i globaliseringens tidsalder: Et forslag til strategi*. Oslo, Norway: Norwegian Language Council. On-line `http://sprakrad.no/Politikk-Fakta/Spraakpolitikk/Norsk_i_hundre_Strategiar/`, Last visited May 21, 2009. xxv, 1, 195

Norwegian Ministry of Culture and Church Affairs. 2008. Mål og meining: Ein heilskapleg norsk språkpolitikk. Stortingsmelding nr. 35 (2007–2008), Norwegian Ministry of Culture and Church Affairs, Oslo, Norway. xxv, 1, 195

Nygaard, Lars. 2006. Frå ordbok til ordnett. Cand.philol.-oppgåve, Universitetet i Oslo, Norway. 4, 34, 192, 194

Oepen, Stephan, Helge Dyvik, Jan Tore Lønning, Erik Velldal, Dorothee Beermann, John Carroll, Dan Flickinger, Lars Hellan, Janne Bondi Johannessen, Paul Meurer, Torbjørn Nordgård, and Victoria Rosén. 2004. Som å kapp-ete med trollet? Towards MRS-based Norwegian—English Machine Translation. In *Proceedings of the 10th International Conference on Theoretical and Methodological Issues in Machine Translation*. Baltimore, MD. 39

Okumura, Akitoshi, and Eduard H. Hovy. 1994. Building Japanese-English dictionary based on ontology for machine translation. In *HLT '94: Proceedings of the Workshop on Human Language Technology*, 141–146. Morristown, NJ, USA: Association for Computational Linguistics. 24, 192

Paşca, Marius A. 2003. *Open domain question answering from large text collections*. CSLI Studies in Computational Linguistics, Center for the Study of Language and Information, Stanford, California: CSLI Publications. 35, 151, 178

Paşca, Marius A., and Sanda M. Harabagiu. 2001. High performance question/answering. In *Proceedings of the 24th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, 366–374. ACM Press. 35

Patterson, David A., and John L. Hennessy. 1998. *Computer Organization & Design: The Hardware/Software Interface*, chap. 9, 710–759. 2nd ed. San Francisco, California, USA: Morgan Kaufmann Publishers, Inc. 180

Pedersen, Bolette Sandford. 2007. Using shallow linguistic analysis to improve search on Danish compounds. *Natural Language Engineering* 13(1):75–90. 23, 67

Pereira, Fernando C. N., and David H. D. Warren. 1980. Definite clause grammars for language analysis—a survey of the formalism and a comparison with augmented transition networks. *Artificial Intelligence* 13(1–2):231–278. 38

Peters, Wim, Piek Vossen, Pedro Díez-Orzas, and Geert Andriaens. 1998. Cross-linguistic alignment of wordnets with an inter-lingual-index. *Computers and the Humanities* 32:221–251. 15

Philpot, Andrew G., Michael Fleischman, and Eduard H. Hovy. 2003. Semi-automatic construction of a general purpose ontology. *Proceedings of the International Lisp Conference. New York, NY. Invited*. 11

Procter, Paul, ed. 1978. *Longman Dictionary of Contemporary English*. Harlow, England: Longman Group Ltd. 24

Pustejovsky, James. 2001. Type construction and the logic of concepts. In *The language of word meaning*, ed. Pierrette Bouillon and Federica Busa, 91–123. Studies in Natural Language Processing, Cambridge, UK: Cambridge University Press. 25

Reiter, Ehud, and Robert Dale. 1997. Building applied natural language generation systems. *Natural Language Engineering* 3:57–87. 16

Resnik, Philip. 1998. WordNet and class-based probabilities. In Fellbaum (1998c), chap. 10, 239–263. 13

Rigau, German. 1994. An experiment on automatic semantic tagging of dictionary senses. In *Proceedings of the International Workshop on the Future of the Dictionary*. Uriage-les-Bains, Grenoble, France. 27

Rigau, German, and Eneko Agirre. 1995. Disambiguating bilingual nominal entries against WordNet. In *Seventh European Summer School in Logic, Language and Information, ESSLLI'95*, 71–82. 26, 146, 147, 192, 196

Russel, Stuart Jonathan, and Peter Norvig, eds. 2003. *Artificial intelligence: A modern approach*. 2nd ed. Prentice Hall Series in Artificial Intelligence, Upper Saddle River, New Jersey, USA: Prentice Hall. 10

Saint-Dizier, Patrick, and Evelyn Viegas, eds. 1995. *Computational lexical semantics*. Studies in natural language processing, Cambridge, UK: Cambridge University Press. 18

Salton, Gerard, A. Wong, and C. S. Yang. 1975. A vector space model for automatic indexing. *Communications of the ACM* 18(11):613–620. 14, 185

Shi, Lei, and Rada Mihalcea. 2005. Putting pieces together: Combining FrameNet, VerbNet and WordNet for robust semantic parsing. In *Proceedings of the 6th International Conference on Computational Linguistics*

and Intelligent Text Processing (CICLing 2005), ed. Alexander Gelbukh, vol. 3406 of *Lecture Notes in Computer Science*, 100–111. Mexico City, Mexico: Springer-Verlag, Berlin, Heidelberg. 81

Simmons, Robert F. 1965. Answering English questions by computer: A survey. *Communications of the ACM* 8(1):53–70. 35

Simonsen, Dag F. 2005. Over the fence—and into English? Reflections on adolescents, academics, linguistic development and language policy in Norway in the early 2000s. In *The Consequences of Mobility: Linguistic and Sociocultural Contact Zones*, ed. Bent Preisler, Anne Fabricius, Hartmut Haberland, Susanne Kjærbeck, and Karen Risager, 249–271. Denmark: Department of Language and Culture, Roskilde University. xxv, 1, 195

Sowa, John F. 2000. *Knowledge representation: Logical, philosophical, and computational foundations*. Pacific Grove, California, USA: Brooks Cole Publishing Co. 10

Statistics Norway. 2009. Population by age, sex, marital status and citizenship. On-line, `http://www.ssb.no/english/subjects/02/01/10/folkemengde_en/`. Last visited May 21, 2009. 1

Stevenson, Mark, and Yorick Wilks. 2001. The interaction of knowledge sources in word sense disambiguation. *Computational Linguistics* 29(3): 321–349. 13

Strzalkowski, Tomek, and Sanda M. Harabagiu, eds. 2006. *Advances in Open Domain Question Answering*, vol. 32 of *Text, Speech and Language Technology*. Dordrecht, The Netherlands: Springer. 35, 208, 214

Sætre, Rune. 2006. GeneTUC: Natural Language Understanding in Medical Text. Ph.D. thesis, Norwegian University of Science and Technology, Trondheim, Norway. 4, 10, 38

Sætre, Rune, Martin Thorsen Ranang, Tonje S. Steigedal, Kamilla Stunes, Kristine Misund, Liv Thommesen, and Astrid Lægreid. 2007. WebProt: Online mining and annotation of biomedical literature using Google. In *Advanced computational methods for biocomputing and bioimaging*, ed. Tuan D. Pham, Hong Yan, and Denis I. Crane. Hauppauge, New York, USA: Nova Science Publishers. 8, 157

Sætre, Rune, Amund Tveit, Martin Thorsen Ranang, Tonje S. Steigedal, Liv Thommesen, Kamilla Stunes, and Astrid Lægreid. 2005. gProt: Annotating protein interactions using Google and Gene Ontology.

In *Proceedings of the 9th International Conference on Knowledge-Based Intelligent Information and Engineering Systems (KES 2005)*, ed. Rajiv Khosla, Robert J. Howlett, and Lakhmi C. Jain, vol. 3683 of *Lecture Notes in Artificial Intelligence*, 1195–1203. Melbourne, Australia: Springer-Verlag, Berlin, Heidelberg. 8, 157

Text Laboratory. 2008a. Oslo-Bergen-taggeren—en grammatisk tagger for bokmål og nynorsk. University of Oslo, On-line, `http://omilia.uio.no/obt/les.html`. Last visited Jan. 29, 2008. 67

———. 2008b. The Oslo corpus of tagged Norwegian texts (bokmål and nynorsk parts). University of Oslo, On-line, `http://www.tekstlab.uio.no/norsk/bokmaal/english.html`. Last visited Jan. 29, 2008. 138

Trask, Robert Lawrence. 1993. *A dictionary of grammatical terms in linguistics*. London, England: Routledge. 13, 17, 22, 76

Tufiş, Dan, Dan Cristea, and Sofia Stamou. 2004. BalkaNet: Aims, methods, results and perspectives. A general overview. *Romanian Journal of Information Science and Technology* 7(1–2):9–43. 25, 192

Verdejo, Felisa, Julio Gonzalo, Anselmo Peñas, Fernando López, and David Fernández. 2000. Evaluating wordnets in Cross-Language Information Retrieval: the ITEM search engine. In *Proceedings of the Second International Conference on Language Resources and Evaluation (LREC-2000)*, 1769–1774. Athens, Greece. 15

Voorhees, Ellen M. 1993. Using WordNet to disambiguate word senses for text retrieval. In *Proceedings of the 16th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, 171–180. ACM Press. 15

———. 1998. Using WordNet for text retrieval. In Fellbaum (1998c), chap. 12, 285–304. 15

Vossen, Piek. 1998. Introduction to EuroWordNet. *Computers and the Humanities* 32:73–89. 15, 24, 192

———. 2004. EuroWordNet: A Multilingual Database of Autonomous and Language-Specific Wordnets Connected via an Inter-Lingual-Index. *International Journal of Lexicography* 17(2):161–173. 15

Vox-Harrap. 1992. *Vox-Harrap's Diccionario Esencial Ingles–Español, Español–Ingles*. Barcelona, Spain: Biblograf S.A. 27

Wangensteen, Boye, ed. 2005. *Bokmålsordboka: Definisjons- og rettskrivningsordbok.* Oslo, Norway: Kunnskapsforlaget. 34, 60

Warren, David H. D., and Fernando C. N. Pereira. 1982. An efficient easily adaptable system for interpreting natural language queries. *American Journal of Computational Linguistics* 8(3–4):110–122. 35

West, Douglas Brent. 2001. *Introduction to graph theory*. 2nd ed. Upper Saddle River, New Jersey, USA: Prentice Hall. 201

Winograd, Terry. 1971. Procedures as a representation for data in a computer program for understanding natural language. MIT AI Technical Report 235, Massachusetts Institute of Technology. Revised Ph.D. dissertation. 35

Woodhouse, Sidney Chawner. 1982. *Latin Dictionary: Latin/English, English/Latin*. 20th ed. London, England: Routledge & Kegan Paul Limited. 6

World Book Encyclopedia. 1987. Chicago, USA: World Book, Inc. 37

Åfarli, Tor A., and Kristin Melum Eide. 2003. *Norsk generativ syntaks*. Novus Forlag, Oslo, Norway. Med tillegg av Lars G. Johnsen, Randi A. Nilsen og Torbjørn Nordgård. 153

Index

Colophon

This dissertation was set in Palatino by the author using LaTeX 2_ε. The author made his best efforts in preparing this document, following guidelines and influences from the following works:

Bringhurst, Robert. 1999. *The Elements of Typographic Style*. 2nd ed. Vancouver, Canada: Hartley & Marks, Publishers.

The Chicago Manual of Style, 15th ed. 2003. Chicago, Illinois, USA: The University of Chicago Press.